LLMs: Emergent Capabilities & Inference Optimization

Lecture 13

18-789

Administrative

- HW3 is out
- Today
  - Paradigm Shift and Emergent Abilities of LLMs
  - Scaling Law
  - Inference Optimization

Transformers, BERT, and GPT-2

- A transformer uses Encoder stack to model input, and uses Decoder stack to model output (using input information from encoder side).
- If we are only interested in training a language model for the input for some other tasks, then we do not need the Decoder of the transformer, that gives us BERT.
- But if we do not have input, we just want to model the “next word”, we can get rid of the Encoder side of a transformer and output “next word” one by one. This gives us GPT.

Encoder Only Model: Bert

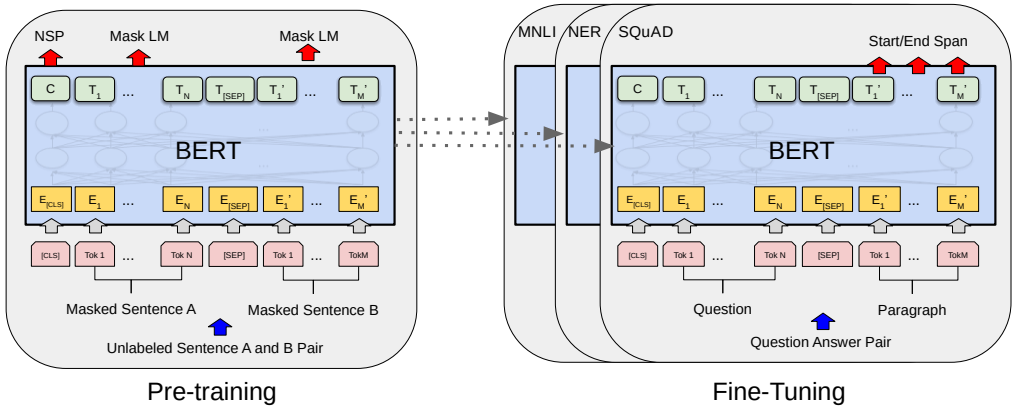
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

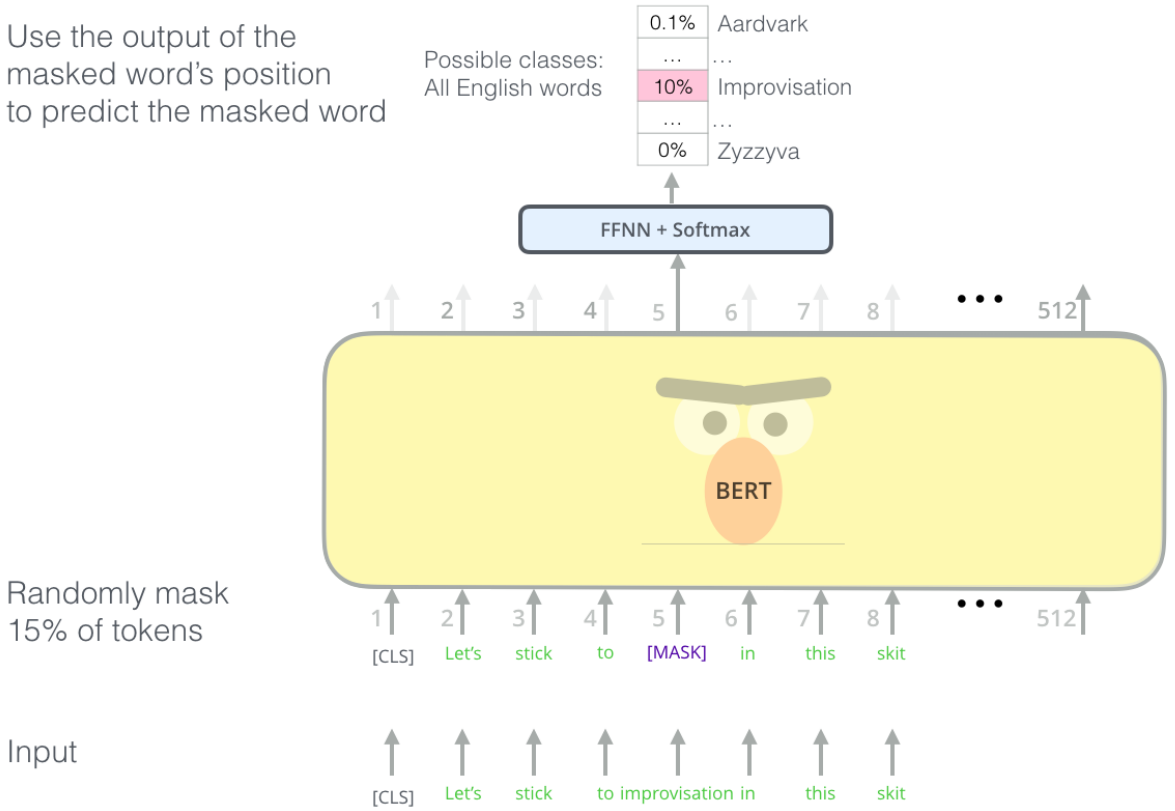
Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

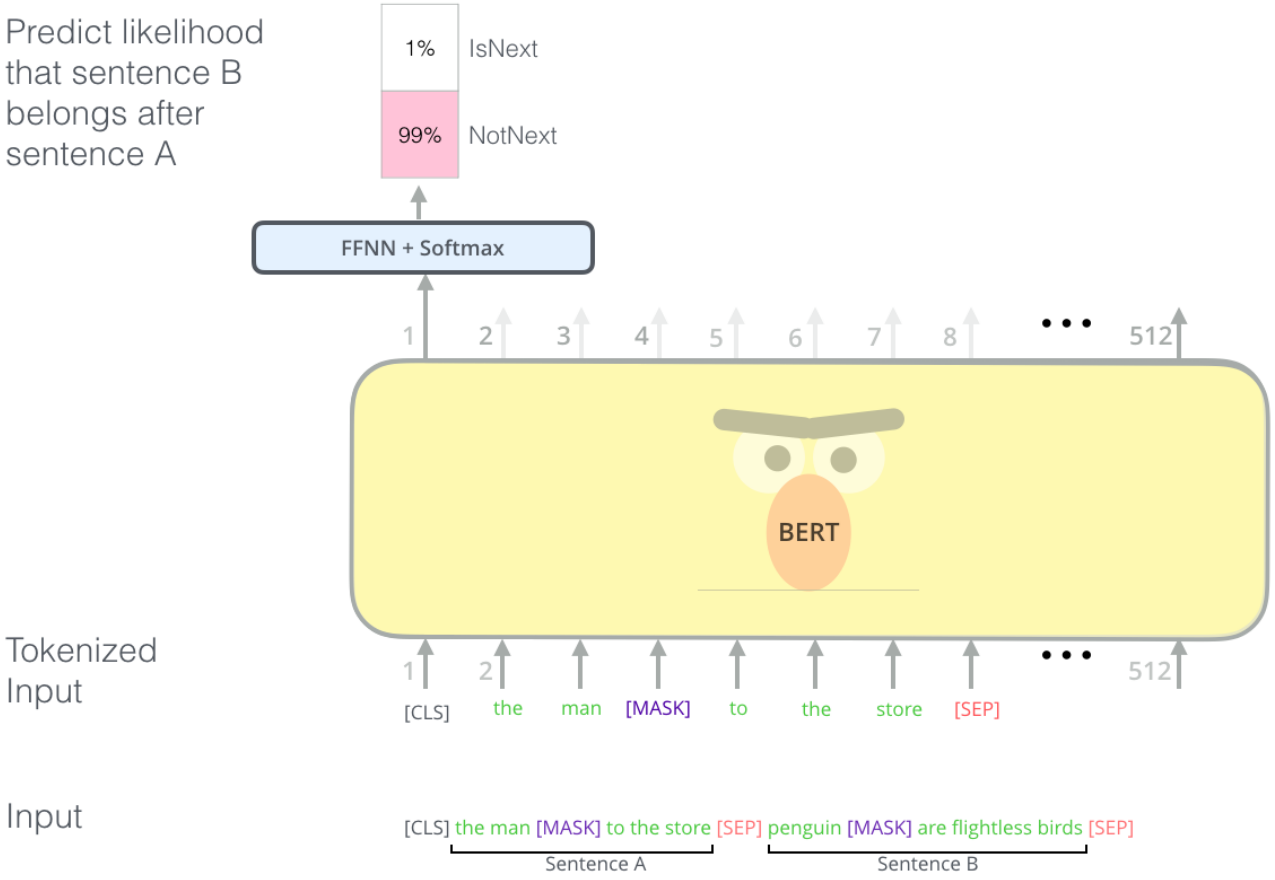


Pretraining Task 1: masked words

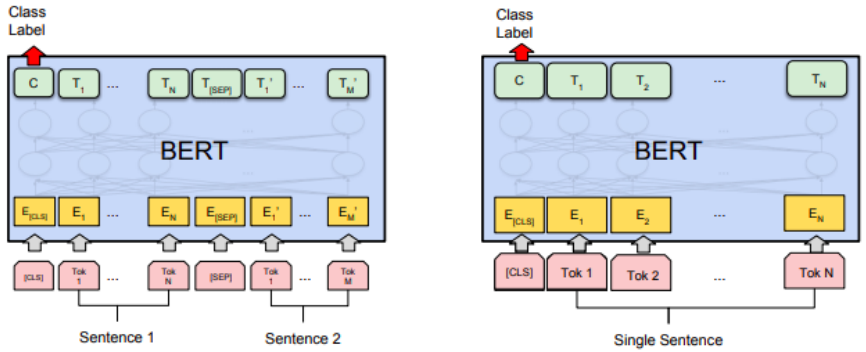


Out of this 15%,
80% are [Mask],
10% random words
10% original words

Pretraining Task 2: two sentences

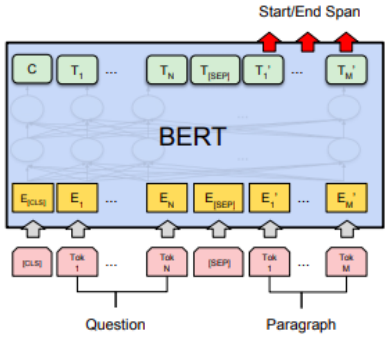


Fine-tuning BERT for other specific tasks

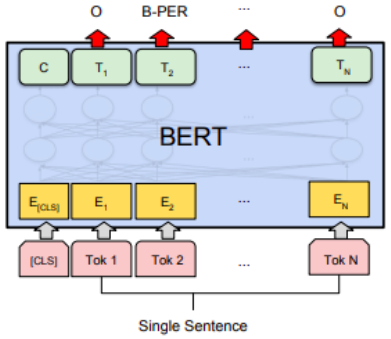


(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

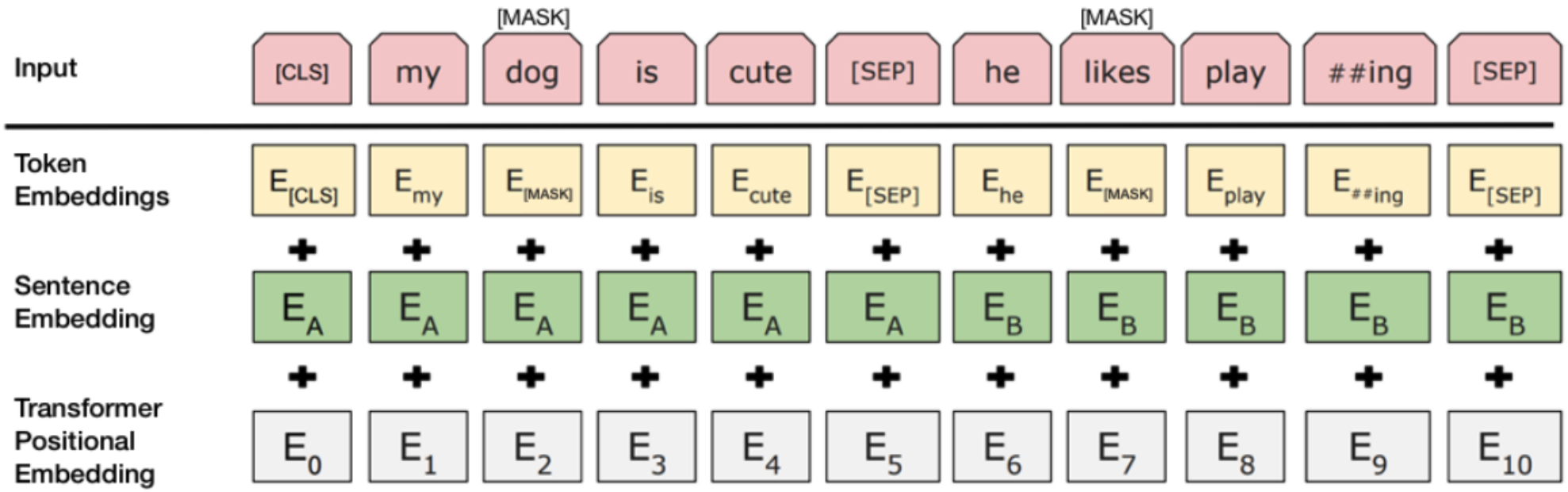


(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Pretraining Task 2: two sentences



Traditional way: Fine-tuning

- Fine-tune the parameters of the pre-trained model for a specific downstream task using a large (thousands to hundreds of thousands) corpus of labeled data.
- Keep training the model via repeated gradient updates.
- Strong performance on many benchmarks.
- Need a new large dataset for each task.
- Potential for poor out-of-distribution generalization
- Potential to explore spurious features of the data

Decoder Only Model: GPT2

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] **Jeffrey Wu** [*1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [**1] **Ilya Sutskever** [**1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al.

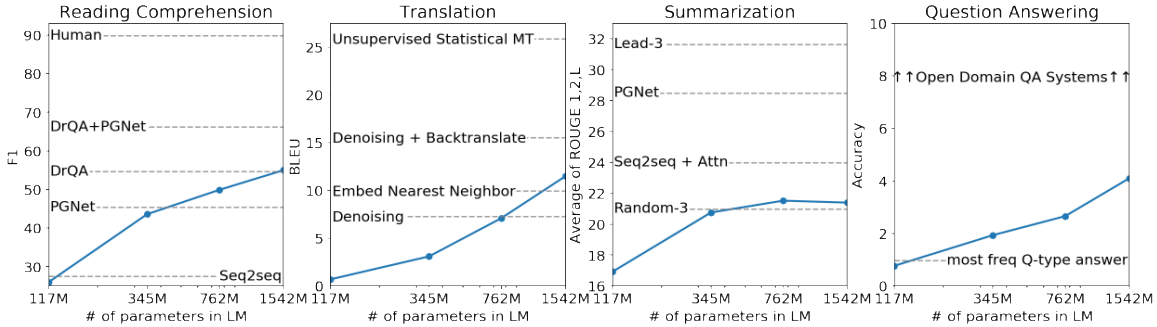
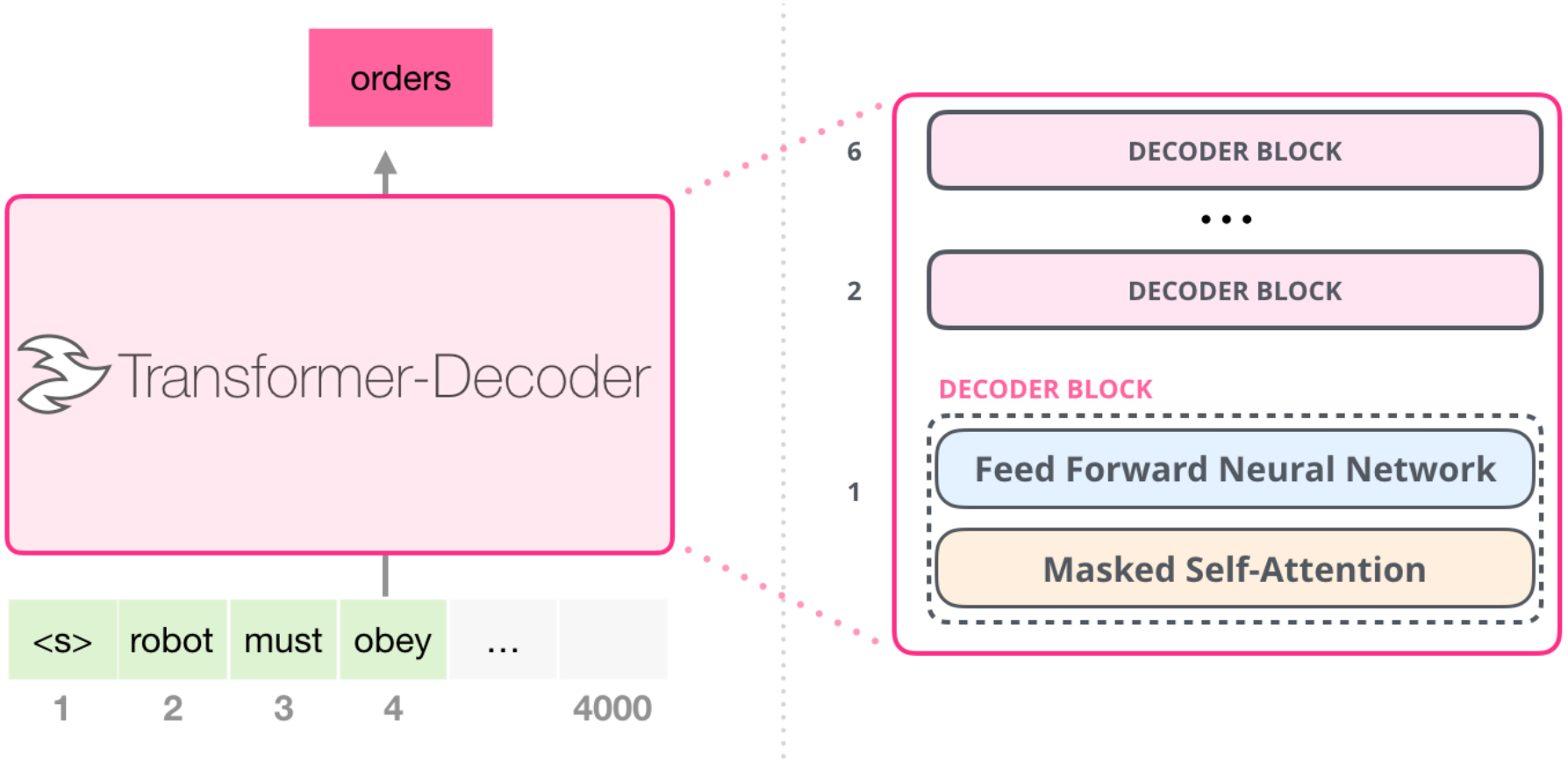


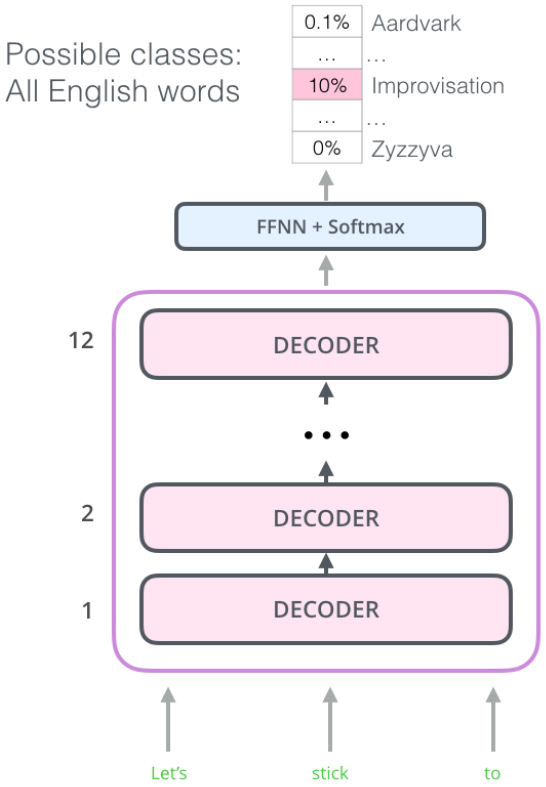
Figure 1. Zero-shot task performance of WebText LMs as a function of model size on many NLP tasks. Reading Comprehension results are on CoQA (Reddy et al., 2018), translation on WMT-14 Fr-En (Artetxe et al., 2017), summarization on CNN and Daily Mail (See et al., 2017), and Question Answering on Natural Questions (Kwiatkowski et al., 2019). Section 3 contains detailed descriptions of each result.

Pretraining Task: next token prediction



Re-use previous computation results: at any step, only need to results of q, k , v related to the new output word, no need to re-compute the others. Additional computation is linear, instead of quadratic.

Pretraining Data



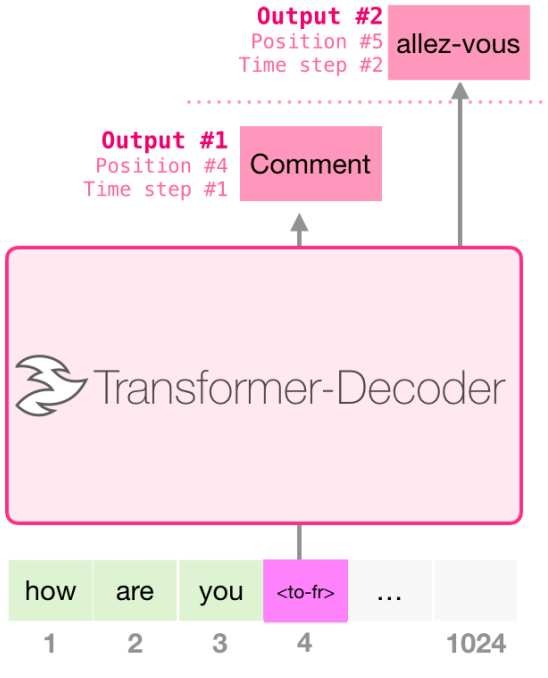
GPT-2 uses unsupervised learning approach to training the language model.

- "Our approach motivates building as large and diverse a dataset as possible in order to collect natural language demonstrations of tasks in as varied of domains and contexts as possible."
- Over 8 million documents for a total of 40 GB of text.

Downstream Tasks

Training Dataset

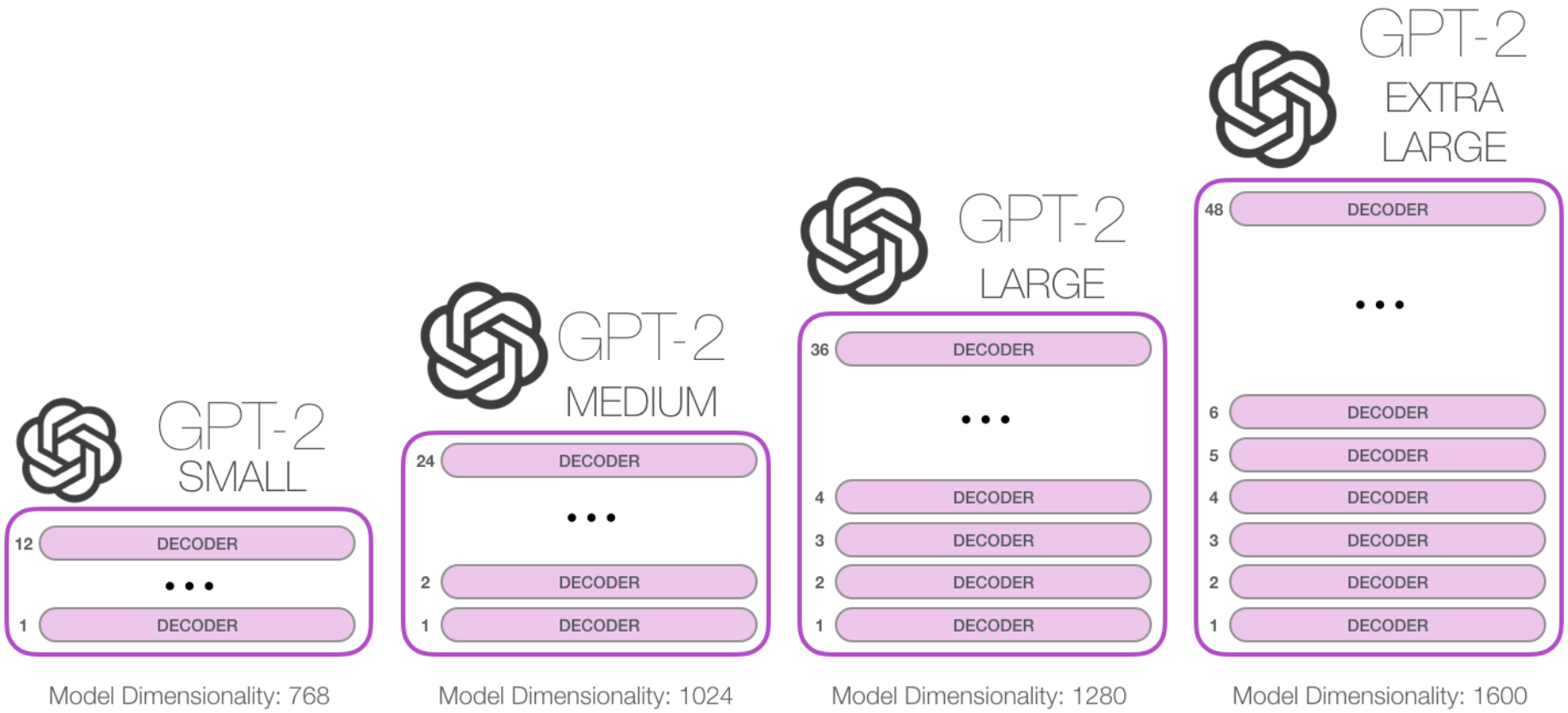
I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



There is no custom training for GPT-2, no separation of pre-training and fine-tuning like BERT.

- Translation
- QA
- Summarization
- Reading Comprehension
- Language Modeling

What makes it work? Scaling



117M parameters 345M 762M 1542M

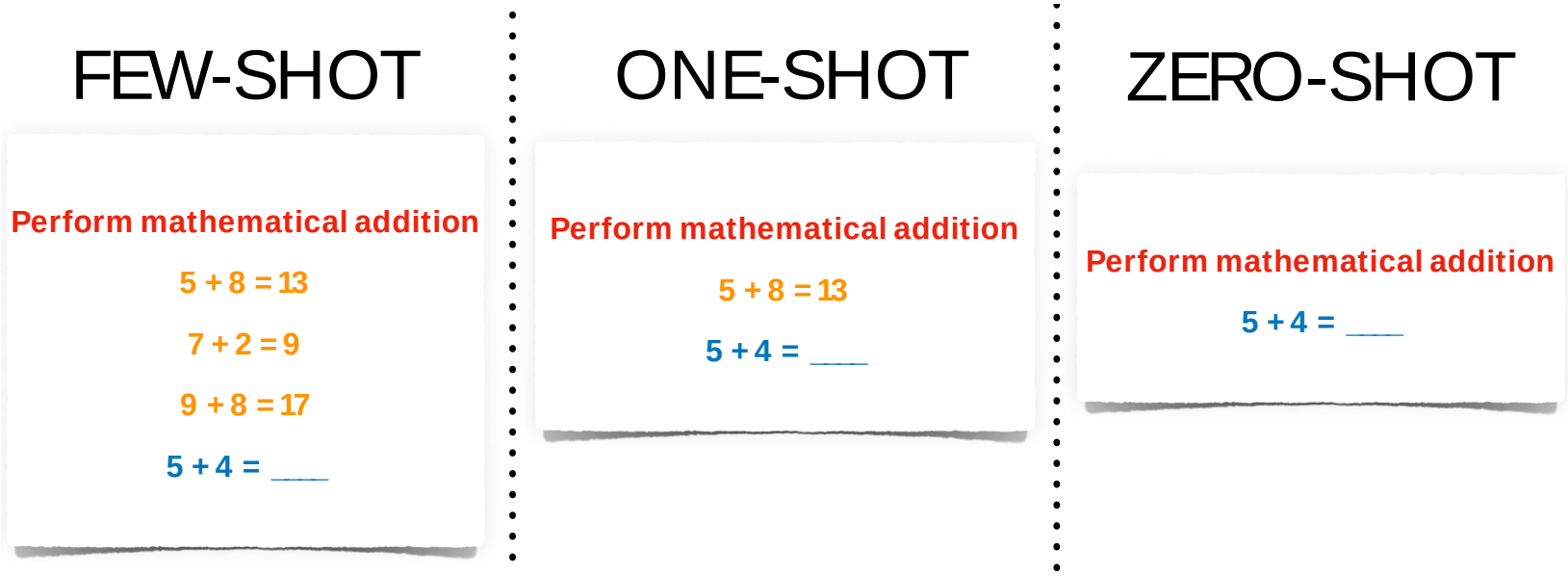
Terminology

1. In-context learning: A frozen LM performs a task only by conditioning on the prompt text.
2. Few-shot in-context learning: (1) The prompt includes examples of the intended behavior, and (2) no examples of the intended behavior were seen in training.
3. Zero-shot in-context learning: (1) The prompt includes no examples of the intended behavior (but it can contain other instructions), and (2) no examples of the intended behavior were seen in training.
4. Emergence: when quantitative changes in a system result in qualitative changes in behavior.
5. Emergent behaviors: abilities that larger models have and smaller models don't

In-context learning

- No training or optimization of the model parameters in the “adaptation step”.Keep training the model via repeated gradient updates.
- Simply give the model a task description as well as none/one/few examples as the input at inference time.
- No gradient updates are performed.

Example


FEW-SHOT
Perform mathematical addition
5 + 8 = 13
7 + 2 = 9
9 + 8 = 17
5 + 4 = ____
ONE-SHOT
Perform mathematical addition
5 + 8 = 13
5 + 4 = ____
ZERO-SHOT
Perform mathematical addition
5 + 4 = ____

Contrast

1. Zero-shot in-context learning
    1. Provides maximum convenience (no task-specific example needed)
    2. Potential for robustness
    3. Potential for avoidance of spurious correlations
    4. Most challenging
    5. Even for humans, it is often hard to understand a task without an example.
2. Few-shot in-context learning
    1. Major reduction in the need for task-specific data.
    2. Reduced potential to learn an overly narrow distribution from a large but narrow fine-tuning dataset.
    3. Still not as good as the fine-tuning SOTA, but competitive (GPT-3).
    4. Still need a few task-specific data

GPT2 – GPT3

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] **Jeffrey Wu** [*1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [**1] **Ilya Sutskever** [**1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al., 2019) and the two most ambitious efforts to date have trained on a total of 10 and 17 (dataset, objective)

Language Models are Few-Shot Learners

Tom B. Brown* **Benjamin Mann*** **Nick Ryder*** **Melanie Subbiah***

Jared Kaplan† **Prafulla Dhariwal** **Arvind Neelakantan** **Pranav Shyam** **Girish Sastry**

Amanda Askell **Sandhini Agarwal** **Ariel Herbert-Voss** **Gretchen Krueger** **Tom Henighan**

Rewon Child **Aditya Ramesh** **Daniel M. Ziegler** **Jeffrey Wu** **Clemens Winter**

Christopher Hesse **Mark Chen** **Eric Sigler** **Mateusz Litwin** **Scott Gray**

Benjamin Chess **Jack Clark** **Christopher Berner**

Sam McCandlish **Alec Radford** **Ilya Sutskever** **Dario Amodei**

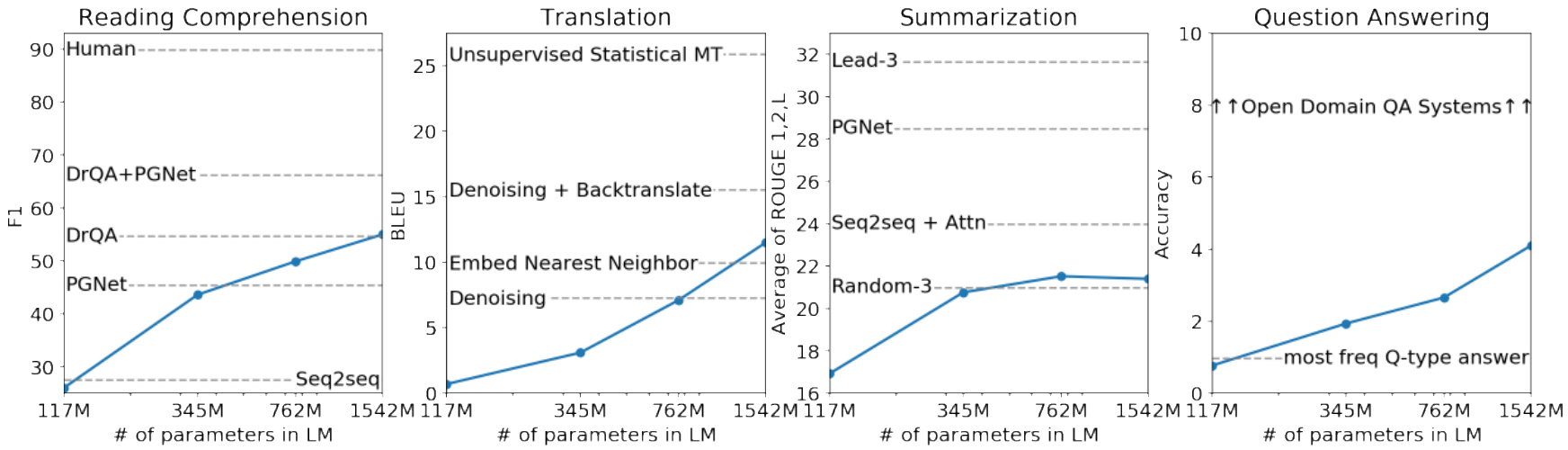
OpenAI

Abstract

Recent work has demonstrated substantial gains on many NLP tasks and benchmarks by pre-training on a large corpus of text followed by fine-tuning on a specific task. While typically task-agnostic in architecture, this method still requires task-specific fine-tuning datasets of thousands or tens of thousands of examples. By contrast, humans can generally perform a new language task from only a few examples or from simple instructions – something which current NLP systems still largely struggle to do. Here we show that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even reaching competitiveness with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic. At the same time, we also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora. Finally, we find that GPT-3 can generate samples of news articles which human evaluators have difficulty distinguishing from articles written by humans. We discuss broader societal impacts of this finding and of GPT-3 in general.

GPT2 – GPT3

The authors “demonstrate that language models begin to learn [question answering, machine translation, reading comprehension, and summarization] tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText.”



GPT2 – GPT3

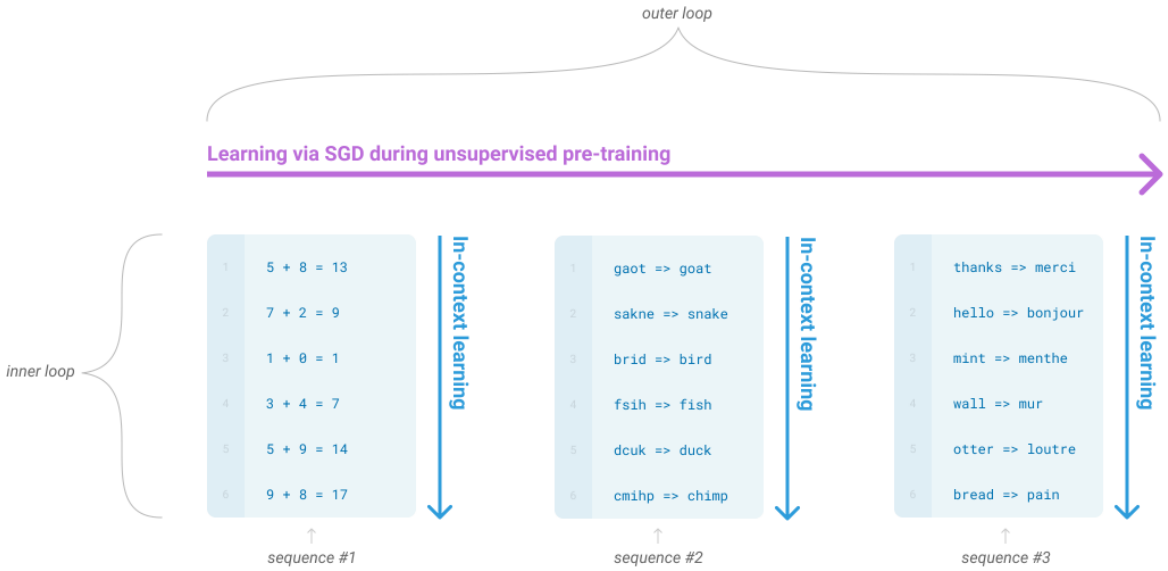


Figure 1.1: Language model meta-learning. During unsupervised pre-training, a language model develops a broad set of skills and pattern recognition abilities. It then uses these abilities at inference time to rapidly adapt to or recognize the desired task. We use the term "in-context learning" to describe the inner loop of this process, which occurs within the forward-pass upon each sequence. The sequences in this diagram are not intended to be representative of the data a model would see during pre-training, but are intended to show that there are sometimes repeated sub-tasks embedded within a single sequence.

GPT2 – GPT3

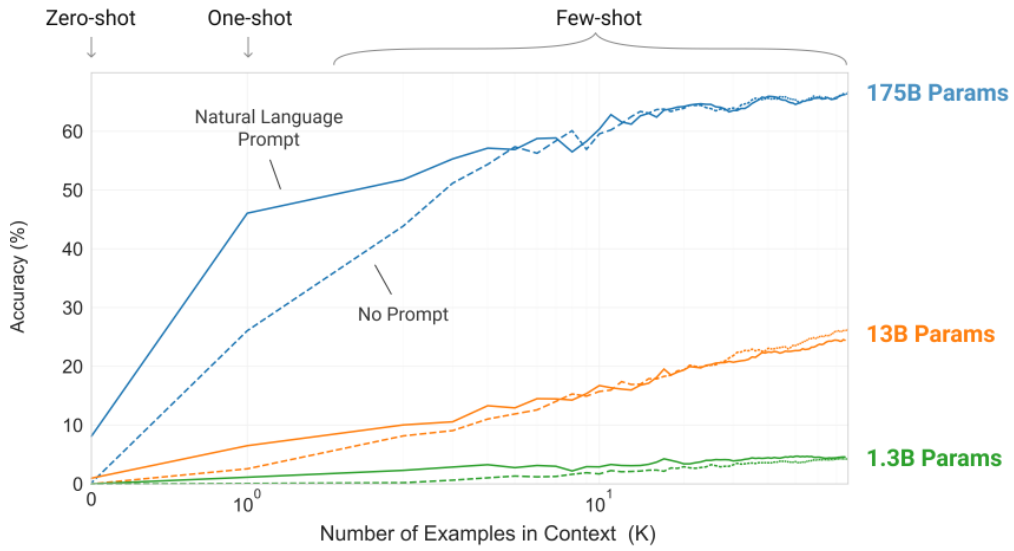


Figure 1.2: Larger models make increasingly efficient use of in-context information. We show in-context learning performance on a simple task requiring the model to remove random symbols from a word, both with and without a natural language task description (see Sec. 3.9.2). The steeper "in-context learning curves" for large models demonstrate improved ability to learn a task from contextual information. We see qualitatively similar behavior across a wide range of tasks.

GPT2 – GPT3

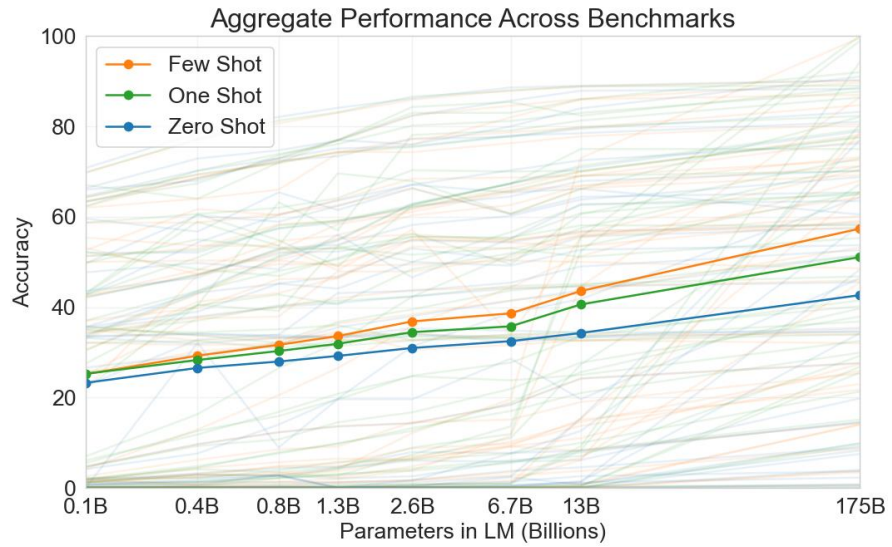


Figure 1.3: Aggregate performance for all 42 accuracy-denominated benchmarks While zero-shot performance improves steadily with model size, few-shot performance increases more rapidly, demonstrating that larger models are more proficient at in-context learning. See Figure 3.8 for a more detailed analysis on SuperGLUE, a standard NLP benchmark suite.

Zero-Shot Learning (Sentiment Classification)

Prompt:

```
Review: Let there be no question:  Alexions owns the best cheeseburger
in the region and they have now for decades. Try a burger on Italian
bread. The service is flawlessly friendly, the food is amazing, and the
wings? Oh the wings... but it's still about the cheeseburger. The
atmosphere is inviting, but you can't eat atmosphere... so go right
now. Grab the car keys... you know you're hungry for an amazing
cheeseburger, maybe some wings, and a cold beer! Easily, hands down,
the best bar and grill in Pittsburgh.

On a 1 to 4 star scale, the reviewer would probably give this
restaurant a
```

Model's Output Probabilities:

P("1" | prompt) = 0.0016
P("2" | prompt) = 0.0032
P("3" | prompt) = 0.012
P("4" | prompt) = 0.78

Template used:

```
Review: <review>

On a 1 to 4 star scale, the reviewer would
probably give this restaurant a <class_label>
```

This template is tried for all possible class labels.

Zero-Shot Learning (Summarization)

Prompt:

```
India's moon rover completes its walk. Scientists analyzing data
looking for signs of frozen water

BEW DELHI -- India’s moon rover has completed its walk on the lunar
surface and been put into sleep mode less than two weeks after its
historic landing near the lunar south pole, India’s space mission said.

“The rover completes its assignments. It is now safely parked and set
into sleep mode," with daylight on that part of the moon coming to an
end, the Indian Space Research Organization said in a statement late
Saturday.

The rover's payloads are turned off and the data it collected has been
transmitted to the Earth via the lander, the statement said.

[...]

Active since the 1960s, India has launched satellites for itself and
other countries, and successfully put one in orbit around Mars in 2014.
India is planning its first mission to the International Space Station
next year, in collaboration with the United States.

TL;DR;
```

Model Generation:

```
India's moon rover has completed its assignments and gone to sleep mode
after just two weeks of being on the lunar surface. The rover has
started transmitting the data it has collected to Earth, but scientists
are unsure if it can survive the extremely cold temperatures on the
moon and wake up in order to conduct another set of missions when the
sun rises again. India successfully landed the rover and underscored
its status as a major tech power and space program.
```

Template used:

```
<title>

<article>

TL;DR;
```

We return the model’s generation from the filled out template.

Few-Shot Learning (Machine Translation)

Prompt:

```
The dog chased a squirrel at the park. = 那只狗在公园里追一只松鼠。

I was late for class. = 我上课迟到了。

The hippopotamus ate my homework. =
```

Model Generation:

河马吃了我的家庭作业。

Prompt with Alternative Template:

```
Translate from English to Chinese.

The dog chased a squirrel at the park. = 那只狗在公园里追一只松鼠。

I was late for class. = 我上课迟到了。

The hippopotamus ate my homework. =
```

Prompt with Alternative Template:

```
Translate from English to Chinese.

English: The dog chased a squirrel at the park.
Chinese: 那只狗在公园里追一只松鼠。

English: I was late for class.
Chinese: 我上课迟到了。

English: The hippopotamus ate my homework.
Chinese:
```

Why does in-context learning work

Transformers Learn In-Context by Gradient Descent

Johannes von Oswald [1 2] **Eyvind Niklasson** [2] **Ettore Randazzo** [2] **João Sacramento** [1]
Alexander Mordvintsev [2] **Andrey Zhmoginov** [2] **Max Vladymyrov** [2]

Abstract

At present, the mechanisms of in-context learning in Transformers are not well understood and remain mostly an intuition. In this paper, we suggest that training Transformers on auto-regressive objectives is closely related to gradient-based meta-learning formulations. We start by providing a simple weight construction that shows the equivalence of data transformations induced by 1) a single linear self-attention layer and by 2) gradient-descent (GD) on a regression loss. Motivated by that construction, we show empirically that when training self-attention-only Transformers on simple regression tasks either the models learned by GD and Transformers show great similarity or, remarkably, the weights found by optimization match the construction. Thus we show how

1. Introduction

In recent years Transformers (TFs; Vaswani et al., 2017) have demonstrated their superiority in numerous benchmarks and various fields of modern machine learning, and have emerged as the *de-facto* neural network architecture used for modern AI (Dosovitskiy et al., 2021; Yun et al., 2019; Carion et al., 2020; Gulati et al., 2020). It has been hypothesised that their success is due in part to a phenomenon called *in-context learning* (Brown et al., 2020; Liu et al., 2021): an ability to flexibly adjust their prediction based on additional data given *in context* (i.e. in the input sequence itself). In-context learning offers a seemingly different approach to few-shot and meta-learning (Brown et al., 2020), but as of today the exact mechanisms of how it works are not fully understood. It is thus of great interest to understand what makes Transformers pay attention to their context, what the mechanisms are, and under which circumstances, they come into play (Chan et al., 2022b; Olsson et al., 2022).

2. Linear self-attention *can* emulate gradient descent on a linear regression task

We start by reviewing a standard multi-head self-attention (SA) layer with parameters θ. A SA layer updates each element e_j of a set of tokens $\{e_1, \ldots, e_N\}$ according to

$$\begin{aligned} e_j &\leftarrow e_j + \mathrm{SA}_\theta(j, \{e_1, \ldots, e_N\}) \\ &= e_j + \sum_h P_h V_h \mathrm{softmax}(K_h^T q_{h,j}) \end{aligned} \tag{1}$$

Why does in-context learning work

Bayesian inference view of in-context learning

Before we get into the Bayesian inference view, let's set up the in-context learning setting.

- **Pretraining distribution** (p): Our main assumption on the structure of pretraining documents is that a document is generated by first sampling a latent concept, and then the document is generated by conditioning on the latent concept. We assume that the pretraining data and LM are large enough that the LM fits the pretraining distribution exactly. Because of this, we will use p to denote both the pretraining distribution and the probability under the LM.
- **Prompt distribution**: In-context learning prompts are lists of IID (independent and identically distributed) training examples concatenated together with one test input. Each example in the prompt is drawn as a sequence conditioned on the same *prompt concept*, which describes the task to be learned.

The process of "locating" learned capabilities can be viewed as Bayesian inference of a prompt concept that every example in the prompt shares. If the model can infer the prompt concept, then it can be used to make the correct prediction on the test example. Mathematically, the prompt provides evidence for the model (p) to sharpen the posterior distribution over concepts, $p(\text{concept} \mid \text{prompt})$. If $p(\text{concept} \mid \text{prompt})$ is concentrated on the prompt concept, the model has effectively "learned" the concept from the prompt.

$$p(\text{output}|\text{prompt}) = \int_{\text{concept}} p(\text{output}|\text{concept}, \text{prompt}) p(\text{concept}|\text{prompt}) d(\text{concept})$$

Ideally, $p(\text{concept} \mid \text{prompt})$ concentrates on the prompt concept with more examples in the prompt so that the prompt concept is "selected" through marginalization.

Why does in-context learning work

- Instances of the task exist in the pre-training data.
  - Example: “TL;DR” is a well-used string on Reddit.
  - Example: Translation data on on the internet
- The few-shot examples “teach” the LLM what format to expect.

Prompt Engineering

Consider the task of classifying the topics of news articles. Which of these prompts do you think would work best?

a) `What is this piece of news regarding?`

b) `What is this article about?`

c) `What is the best way to describe this article?`

d) `What is the most accurate label for this news article?`

e) They should all perform about the same.

Prompt Engineering

Consider the task of classifying the topics of news articles. Which of these prompts do you think would work best?

accuracies according to OPT-175B

a) `What is this piece of news regarding?` 40.9%

b) `What is this article about?` 52.4%

c) `What is the best way to describe this article?` 68.2%

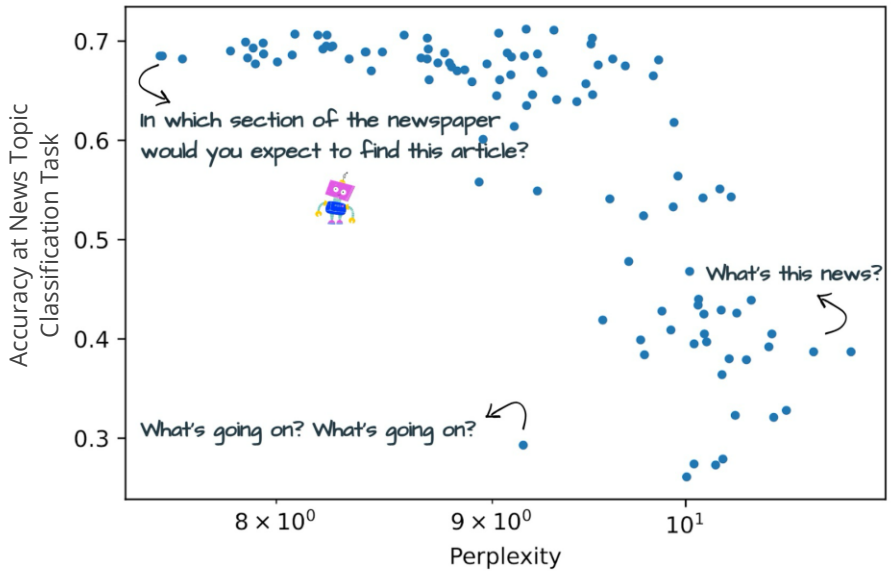
d) `What is the most accurate label for this news article?` 71.2%

e) They should all perform about the same.

Prompts which are perceptually equivalent to humans can result in radically different performance.

What matters in prompt selection?

- Prompts which are perceptually equivalent to humans can result in radically different performance.
- Prompt performance is correlated with the extent to which the model is familiar with the language the prompt contains.



What matters in prompt selection?

Consider the task of labeling movie reviews as positive or negative sentiment. Which of the following prompts should work better?

	Prompt (test input not shown)
A	Review: the whole thing 's fairly lame , making it par for the course for disney sequels . Answer: Negative Review: this quiet , introspective and entertaining indepen-dent is worth seeking . Answer: Positive
B	Review: this quiet , introspective and entertaining indepen-dent is worth seeking . Answer: Positive Review: the whole thing 's fairly lame , making it par for the course for disney sequels . Answer: Negative
C	**They should perform about the same.**

What matters in prompt selection?

Consider the task of labeling movie reviews as positive or negative sentiment. Which of the following prompts should work better?

	Prompt (test input not shown)	**Acc.**
A	Review: the whole thing 's fairly lame , making it par for the course for disney sequels . Answer: Negative Review: this quiet , introspective and entertaining indepen-dent is worth seeking . Answer: Positive	88.5%
B	Review: this quiet , introspective and entertaining indepen-dent is worth seeking . Answer: Positive Review: the whole thing 's fairly lame , making it par for the course for disney sequels . Answer: Negative	51.3%
C	They should perform about the same.	

What matters in prompt selection?

- Prompts which are perceptually equivalent to humans can result in radically different performance.
- Prompt performance is correlated with the extent to which the model is familiar with the language the prompt contains.
- Few-shot example choice and ordering make a huge difference in performance.
- LLMs can be biased toward answers which occur more frequently in the prompt.

What matters in prompt selection?

Calibrate Before Use:
Improving Few-Shot Performance of Language Models

Tony Z. Zhao [*1] Eric Wallace [*1] Shi Feng [2] Dan Klein [1] Sameer Singh [3]

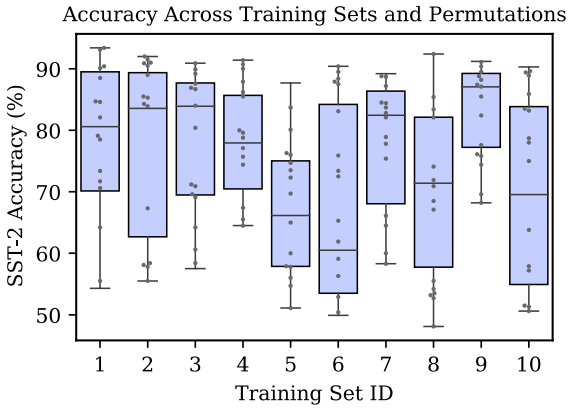
Abstract

GPT-3 can perform numerous tasks when provided a natural language prompt that contains a few training examples. We show that this type of few-shot learning can be unstable: the choice of prompt format, training examples, and even the order of the training examples can cause accuracy to vary from near chance to near state-of-the-art. We demonstrate that this instability arises from the bias of language models towards predicting certain answers, e.g., those that are placed near the end of the prompt or are common in the pre-training data. To mitigate this, we first estimate the model's bias towards each answer by asking for its prediction when given the training prompt and a content-free test input such as "N/A". We then fit calibration parameters that cause the prediction for this input to be uniform across answers. On a diverse set of tasks, this *contextual calibration* procedure substantially improves GPT-3 and GPT-2's average accuracy (up to 30.0% absolute) and reduces variance across different choices of the prompt.

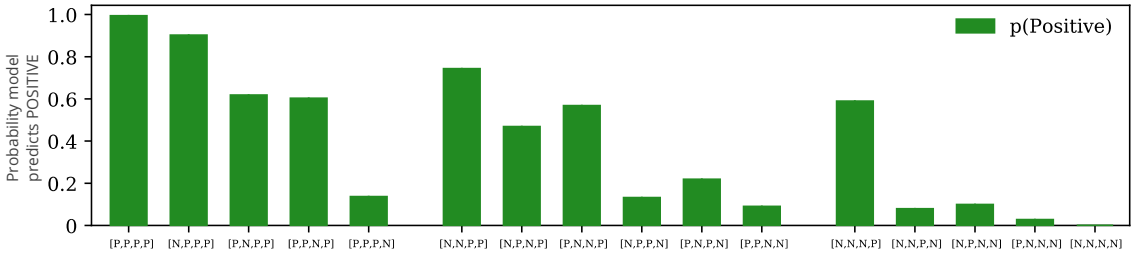
where the first two lines correspond to two training examples and the last line is a test example. To make predictions, the model predicts whether the subsequent token is more likely to be the word "Positive" or "Negative".

This style of few-shot "in-context" learning is interesting because it shows that the model can learn without parameter updates. And, more importantly, it has numerous practical advantages over the now-standard approach of finetuning (Radford et al., 2018; Devlin et al., 2019). First, it allows practitioners to "rapidly prototype" NLP models: changing the prompt *immediately* leads to a new model. Second, it provides a fully natural language interface to a machine learning model, which allows users—even those without technical expertise—to create NLP systems. Finally, since in-context learning reuses the same model for each task, it reduces memory requirements and system complexity when serving many different tasks.

However, despite these promises, we show that GPT-3's accuracy can be highly unstable across different prompts (Section 3). A prompt contains three components: a format, a set of training examples, and a permutation (ordering) for those examples. We show that different choices for these factors can lead to highly different accuracies, e.g., changing the permutation of the training examples in a sentiment



Each box plot represents all permutations of a set of 4 train set examples.



What matters in prompt selection?

- Prompts which are perceptually equivalent to humans can result in radically different performance.
- Prompt performance is correlated with the extent to which the model is familiar with the language the prompt contains.
- Few-shot example choice and ordering make a huge difference in performance.
- LLMs can be biased toward answers which occur more frequently in the prompt.
- Labels can be wrong and it doesn’t matter.

What matters in prompt selection?

Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?

Sewon Min[1,2] Xinxi Lyu[1] Ari Holtzman[1] Mikel Artetxe[2]
Mike Lewis[2] Hannaneh Hajishirzi[1,3] Luke Zettlemoyer[1,2]
[1]University of Washington [2]Meta AI [3]Allen Institute for AI
{sewon, alrope, ahai, hannaneh, lsz}@cs.washington.edu
{artetxe, mikelewis}@meta.com

Abstract

Large language models (LMs) are able to in-context learn—perform a new task via inference alone by conditioning on a few input-label pairs (demonstrations) and making predictions for new inputs. However, there has been little understanding of *how* the model learns and *which* aspects of the demonstrations contribute to end task performance. In this paper, we show that ground truth demonstrations are in fact not required—randomly replacing labels in the demonstrations barely hurts performance on a range of classification and multi-choce tasks, consistently over 12 different models including GPT-3. Instead, we find that other aspects of the demonstrations are the key drivers of end task performance, including the fact that they provide a few examples of (1) the label space, (2) the distribution of the input text, and (3) the overall format of the sequence. Together, our analysis provides

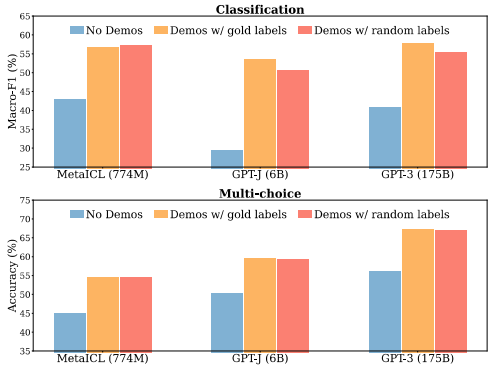
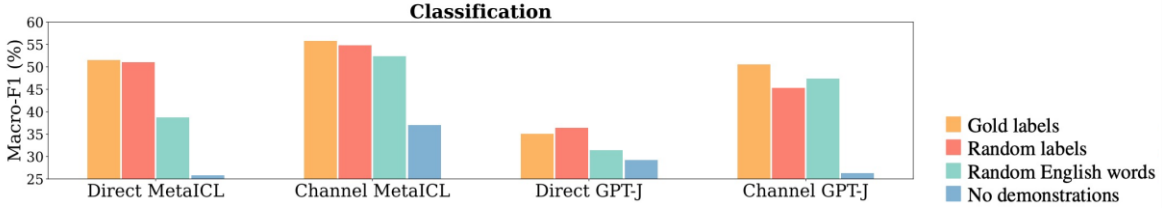


Figure 1: Results in classification (top) and multi-choice tasks (bottom), using three LMs with varying size. Reported on six datasets on which GPT-3 is evaluated; the channel method is used. See Section 4 for the full results. In-context learning performance drops only marginally when labels in the demonstrations are replaced by random labels.



How much does having more exemplars help?

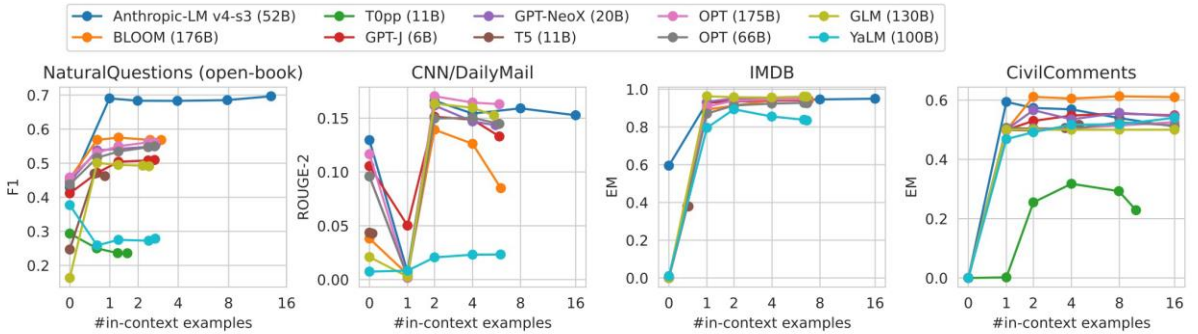
Holistic Evaluation of Language Models

Percy Liang[†], Rishi Bommasani[†], Tony Lee[†], Dimitris Tsipras[‡], Dilara Soylu[‡], Michihiro Yasunaga[‡], Yian Zhang[‡], Deepak Narayanan[‡], Yuhuai Wu[‡], Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, Yuta Koreeda
pliang@cs.stanford.edu, nlprishi@stanford.edu, tonyhlee@stanford.edu
Center for Research on Foundation Models (CRFM)
Institute for Human-Centered Artificial Intelligence (HAI)
Stanford University

Reviewed on OpenReview: *https://openreview.net/forum?id=iO4LZibEqW*

Abstract

Language models (LMs) are becoming the foundation for almost all major language technologies, but their capabilities, limitations, and risks are not well understood. We present Holistic Evaluation of Language Models (HELM) to improve the transparency of language models. First, we taxonomize the vast space of potential scenarios (i.e. use cases) and metrics (i.e. desiderata) that are of interest for LMs. Then we select a broad subset based on coverage and feasibility, noting what's missing or underrepresented (e.g. question answering for neglected English dialects, metrics for trustworthiness). Second, we adopt a multi-metric approach: We measure 7 metrics (accuracy, calibration, robustness, fairness, bias, toxicity, and efficiency) for *each* of 16 core scenarios to the extent possible (87.5% of the time), ensuring that metrics beyond accuracy don't fall to the wayside, and that trade-offs across models and metrics are clearly exposed. We also perform 7 targeted evaluations, based on 26 targeted scenarios, to more deeply analyze specific aspects (e.g. knowledge, reasoning, memorization/copyright, disinformation). Third, we conduct a large-scale evaluation of 30 prominent language models (spanning open, limited-access, and closed models) on all 42 scenarios, including 21 scenarios that were not previously used in mainstream LM evaluation. Prior to HELM, models on average were evaluated on just 17.9% of the core HELM scenarios, with some prominent models not sharing a single scenario in common. We improve this to 96.0%: now all 30 models have been densely benchmarked on a set of core scenarios and metrics under standardized conditions. Our evaluation surfaces 25 top-level findings concerning the interplay between different scenarios, metrics, and models. For full transparency, we release all raw model prompts and completions publicly[1] for further analysis, as well as a general modular toolkit for easily adding new scenarios, models, metrics, and prompting strategies.[2] We intend for HELM to be a living benchmark for the community, continuously updated with new scenarios, metrics, and models.



Chain-of-Thought Prompting

Intuition: An LLM will be better able to perform tasks (especially reasoning-based ones) if it is made to break down the task into multiple small steps.

Main idea: each of the exemplars in your few-shot prompt contains logic showing how to solve the task.

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei Xuezhi Wang Dale Schuurmans Maarten Bosma
Brian Ichter Fei Xia Ed H. Chi Quoc V. Le Denny Zhou

Google Research, Brain Team
{jasonwei,dennyzhou}@google.com

Abstract

We explore how generating a *chain of thought*—a series of intermediate reasoning steps—significantly improves the ability of large language models to perform complex reasoning. In particular, we show how such reasoning abilities emerge naturally in sufficiently large language models via a simple method called *chain-of-thought prompting*, where a few chain of thought demonstrations are provided as exemplars in prompting.

Experiments on three large language models show that chain-of-thought prompting improves performance on a range of arithmetic, commonsense, and symbolic reasoning tasks. The empirical gains can be striking. For instance, prompting a PaLM 540B with just eight chain-of-thought exemplars achieves state-of-the-art accuracy on the GSM8K benchmark of math word problems, surpassing even finetuned GPT-3 with a verifier.

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Step-by-step demonstration

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Step-by-step answer

Zero-Shot Chain-of-Thought Prompting

Large Language Models are Zero-Shot Reasoners

Takeshi Kojima
The University of Tokyo
t.kojima@weblab.t.u-tokyo.ac.jp

Shixiang Shane Gu
Google Research, Brain Team

Machel Reid
Google Research*

Yutaka Matsuo
The University of Tokyo

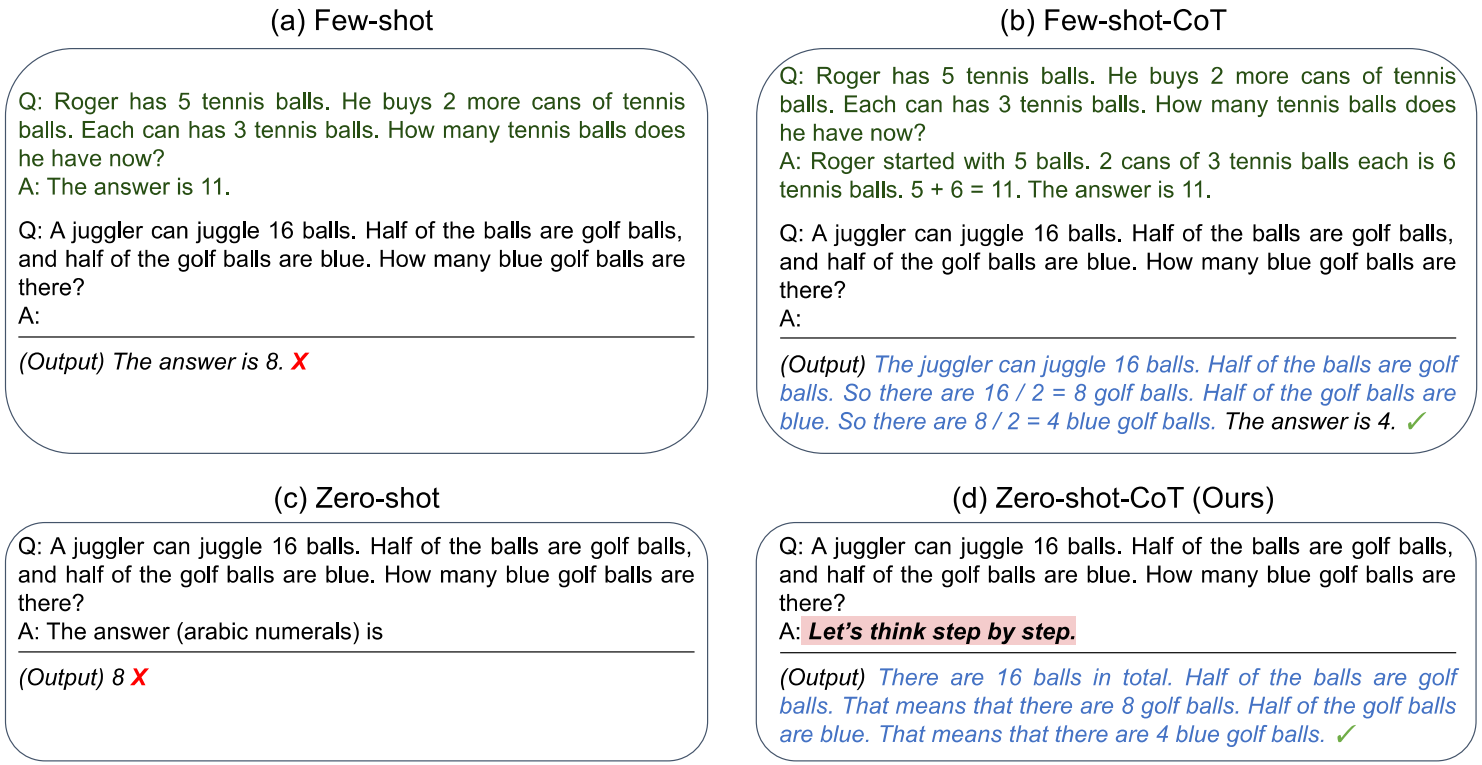
Yusuke Iwasawa
The University of Tokyo

Abstract

Pretrained large language models (LLMs) are widely used in many sub-fields of natural language processing (NLP) and generally known as excellent *few-shot* learners with task-specific exemplars. Notably, chain of thought (CoT) prompting, a recent technique for eliciting complex multi-step reasoning through step-by-step answer examples, achieved the state-of-the-art performances in arithmetics and symbolic reasoning, difficult *system-2* tasks that do not follow the standard scaling laws for LLMs. While these successes are often attributed to LLMs' ability for few-shot learning, we show that LLMs are decent *zero-shot* reasoners by simply adding "Let's think step by step" before each answer. Experimental results demonstrate that our Zero-shot-CoT, using the same single prompt template, significantly outperforms zero-shot LLM performances on diverse benchmark reasoning tasks including arithmetics (MultiArith, GSM8K, AQUA-RAT, SVAMP), symbolic reasoning (Last Letter, Coin Flip), and other logical reasoning tasks (Date Understanding, Tracking Shuffled Objects), without any hand-crafted few-shot examples, e.g. increasing the accuracy on MultiArith from 17.7% to 78.7% and GSM8K from 10.4% to 40.7% with large-scale InstructGPT model (text-davinci-002), as well as similar magnitudes of improvements with another off-the-shelf large model, 540B parameter PaLM. The versatility of this single prompt across very diverse reasoning tasks hints at untapped and understudied fundamental *zero-shot* capabilities of LLMs, suggesting high-level, multi-task broad cognitive capabilities may be extracted by simple prompting. We hope our work not only serves as the minimal strongest zero-shot baseline for the challenging reasoning benchmarks, but also highlights the importance of carefully exploring and analyzing the enormous zero-shot knowledge hidden inside LLMs before crafting finetuning datasets or few-shot exemplars.

Main idea: We don't need any exemplars! Just append the string "Let's think step by step." to the end of the prompt.

Zero-Shot Chain-of-Thought Prompting



Zero-Shot Chain-of-Thought Prompting

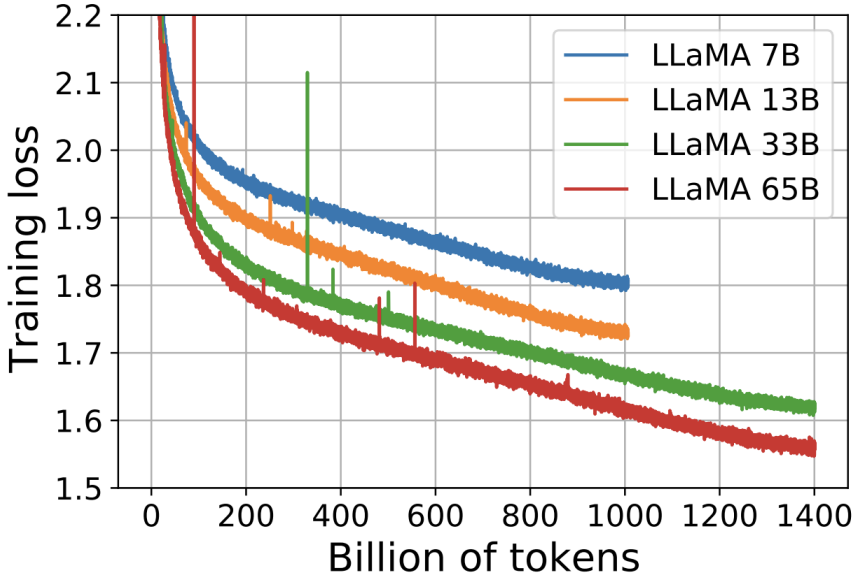
Advantages over chain-of-thought (CoT) method:

- The single fixed instruction “Let’s think step by step” works over a large variety of different tasks.
- CoT performance degrades when there is misalignment between the example question types in the prompt and the actual task question.
- In summary, multi-step prompting requires less human time spent on prompt engineering.

What makes it work?

"The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin."

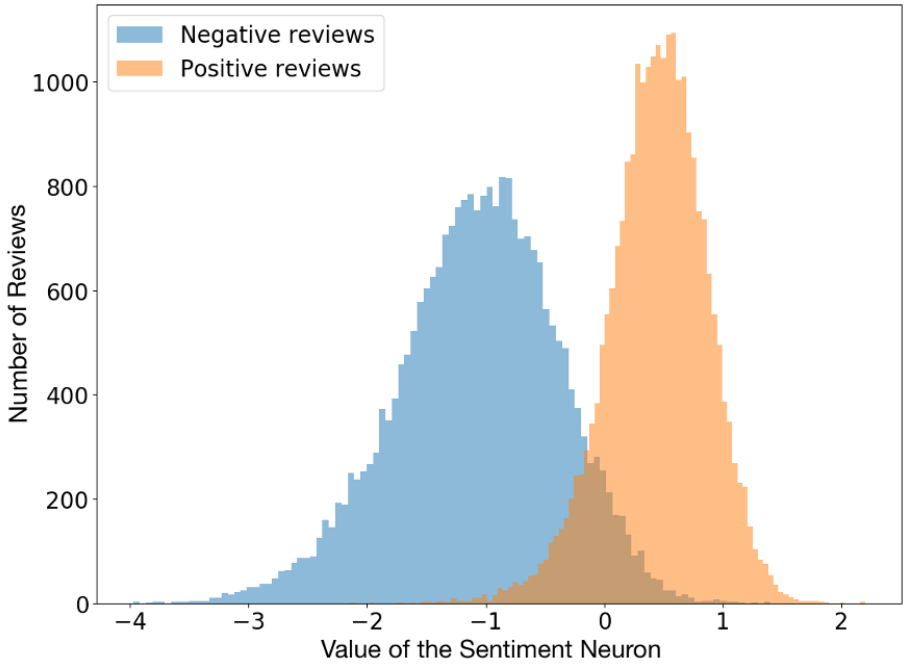
-- The Bitter Lesson, Richard Sutton 2019



We want to model data distribution better by adding more compute.

What makes it work? Model Scaling

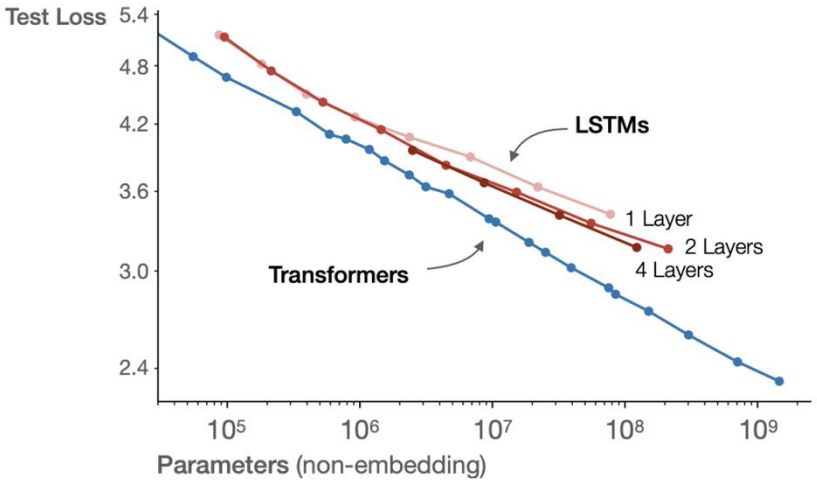
LSTM learns a sentiment neuron after training to predict the next word on a large amount of Amazon reviews.

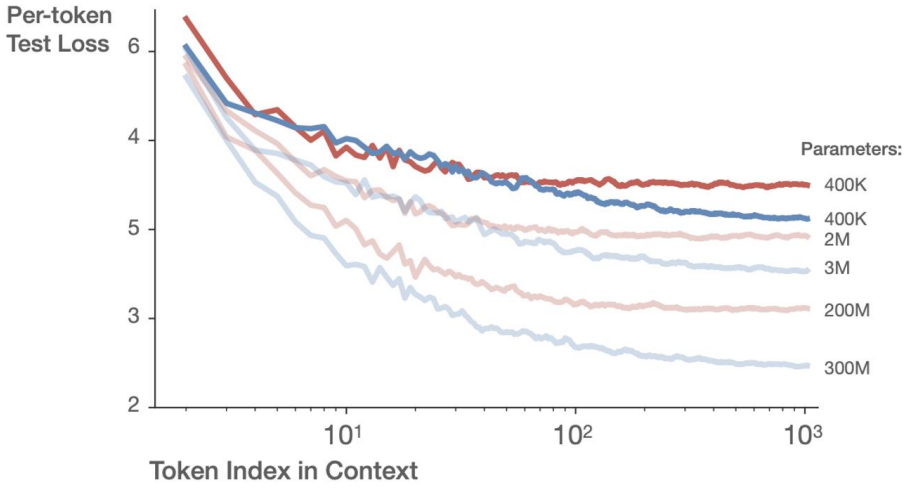


Radford, Alec, Rafal Jozefowicz, and Ilya Sutskever. "Learning to generate reviews and discovering sentiment." arXiv 2017.

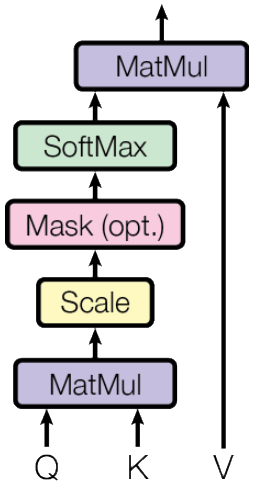
What makes it work? Model Scaling

LSTM learns a sentiment neuron after training to predict the next word on a large amount of Amazon reviews.



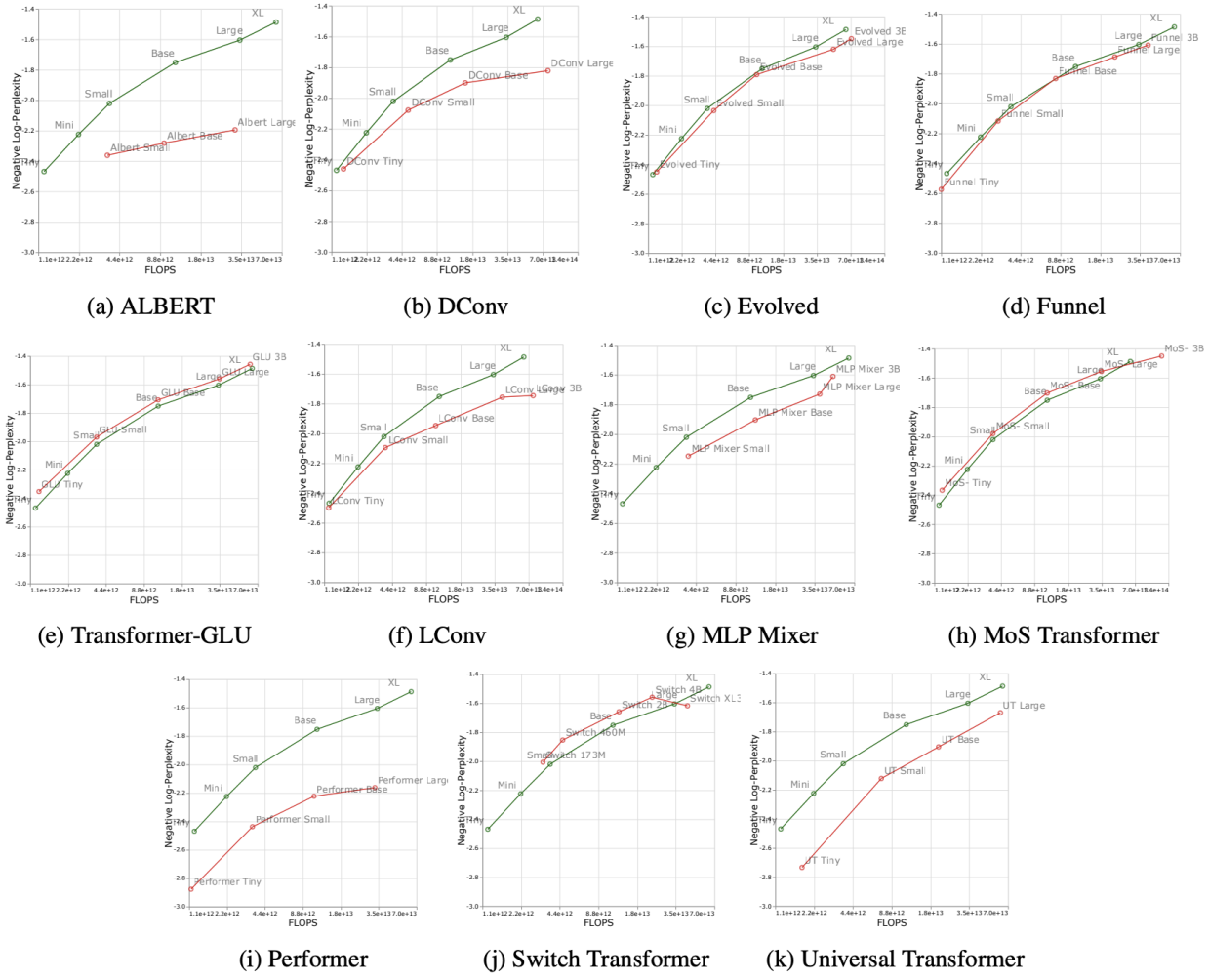


Scaled Dot-Product Attention



Scaling laws for neural language models. Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020)

What makes it work? Model Scaling



(a) ALBERT (b) DConv (c) Evolved (d) Funnel

(e) Transformer-GLU (f) LConv (g) MLP Mixer (h) MoS Transformer

(i) Performer (j) Switch Transformer (k) Universal Transformer

- Upstream Negative Log-Perplexity of vanilla Transformer compared to other models.
- Transformer outperforms other models.

Tay, Y., Dehghani, M., Abnar, S., Chung, J., … & Metzler, D. Scaling laws vs model architectures: How does inductive bias influence scaling? (2022).

What makes it work? Model Scaling

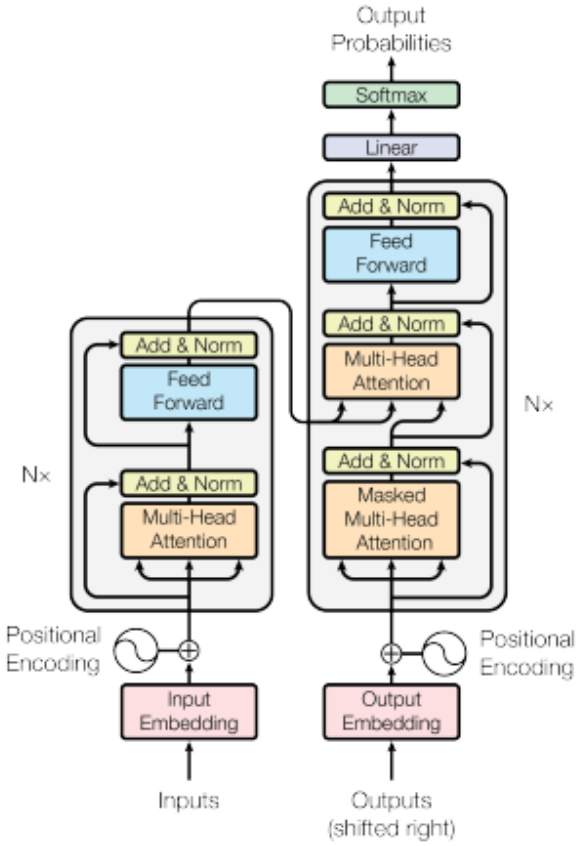


Figure 1: The Transformer - model architecture.

Decoder only model leverages data in the most efficient way.

What makes it work? Model and Data Scaling

Parameter count is N, token count is D, what is the compute cost?

- C = 6ND(1+s/6d): Compute increases with more data and larger model
- LLaMA (s << 6d): C = 6ND = 6 ∗ 7 billion ∗ 2 trillion = 8.4 × 10^22 FLOPs

Shall I allocate more compute to data or model?

$$N_{opt}(C), D_{opt}(C) = \underset{N,D \text{ s.t. } \mathrm{FLOPs}(N,D)=C}{\operatorname{argmin}} L(N, D)$$

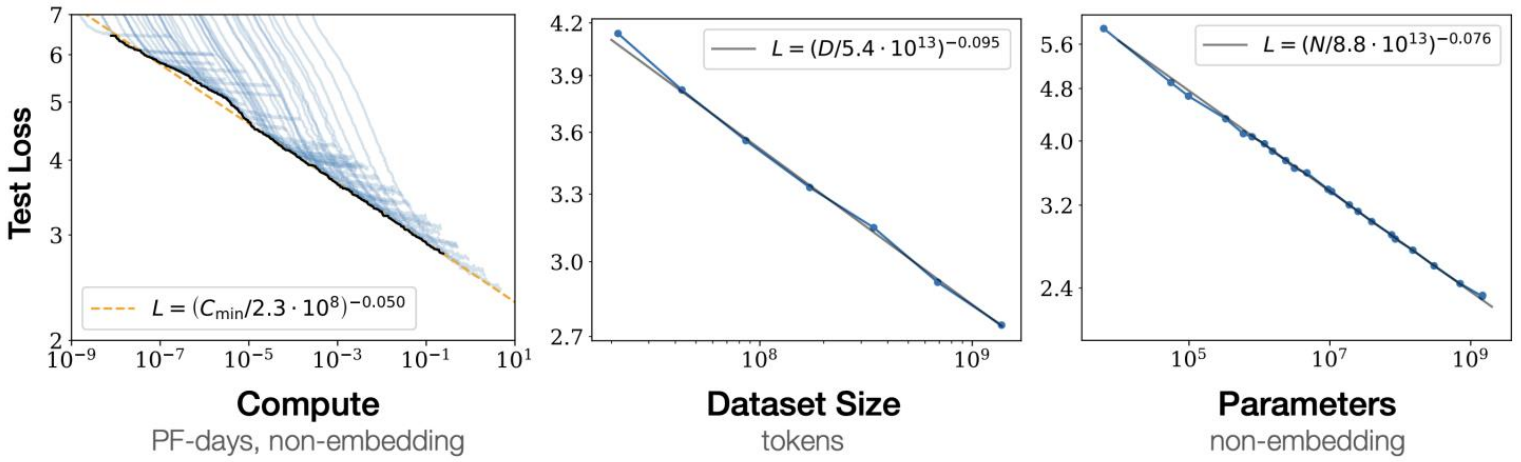
Scaling laws for neural language models. Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020)

Compute Scaling

Empirical performance has a power-law relationship with each individual factor.

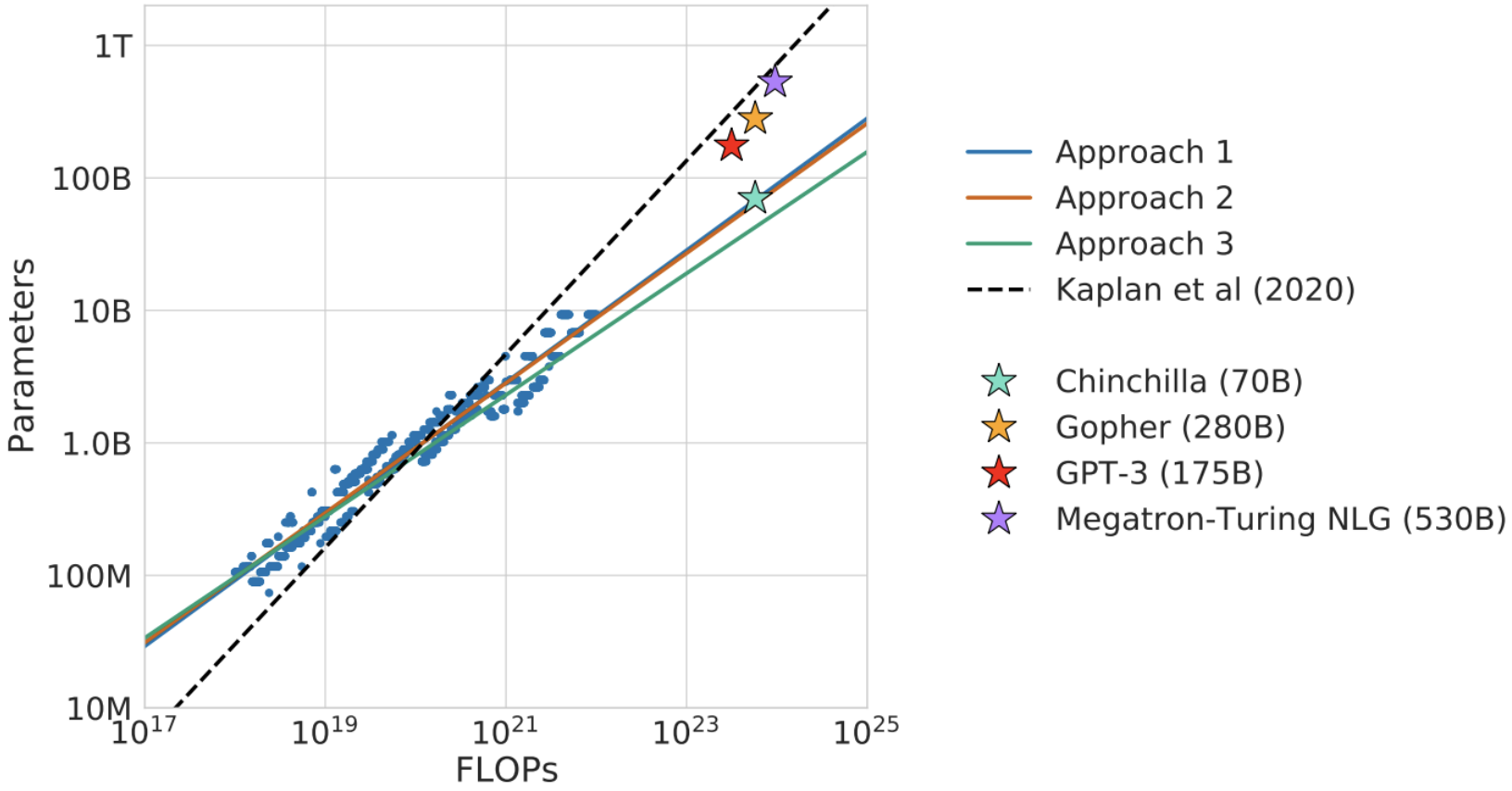
- Train model of different sizes and number of tokens (light blue lines)
- Pick minimal loss (black line)
- Run linear regression on log – log

$$N_{opt} \propto C^a, D_{opt} \propto C^b$$



Scaling laws for neural language models. Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020)

Chinchilla Scaling



"Training Compute-Optimal Large Language Models." (2022).

Chinchilla Scaling

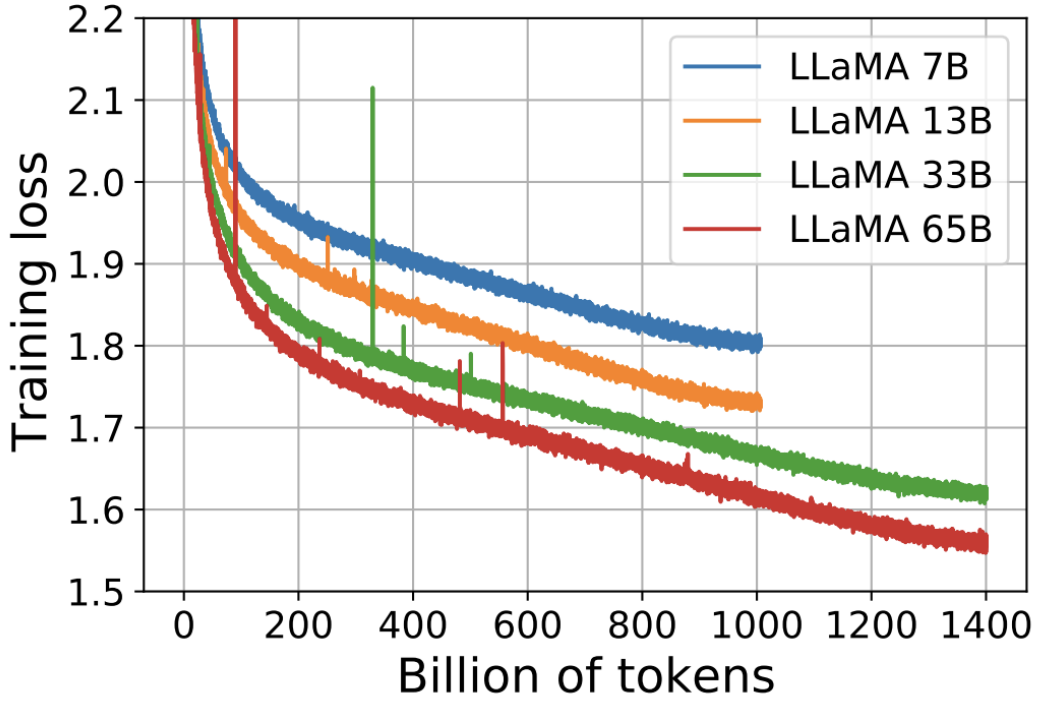
- Estimated optimal training tokens: about 20 times number of parameters
- It's best to double the estimate: tokens = 40x parameters

Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

"Training Compute-Optimal Large Language Models." (2022).

Inference Optimal

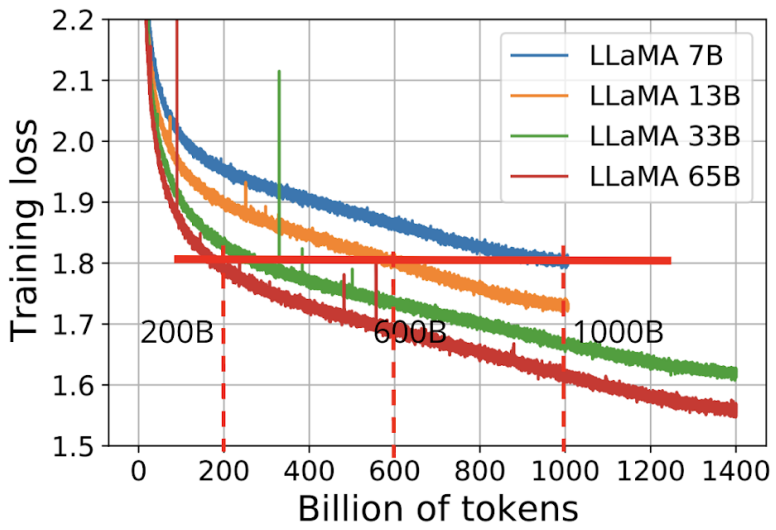
Train small model and more tokens to achieve better inference
E.g. Llama 7b is trained on 7 times of Chinchilla optimal

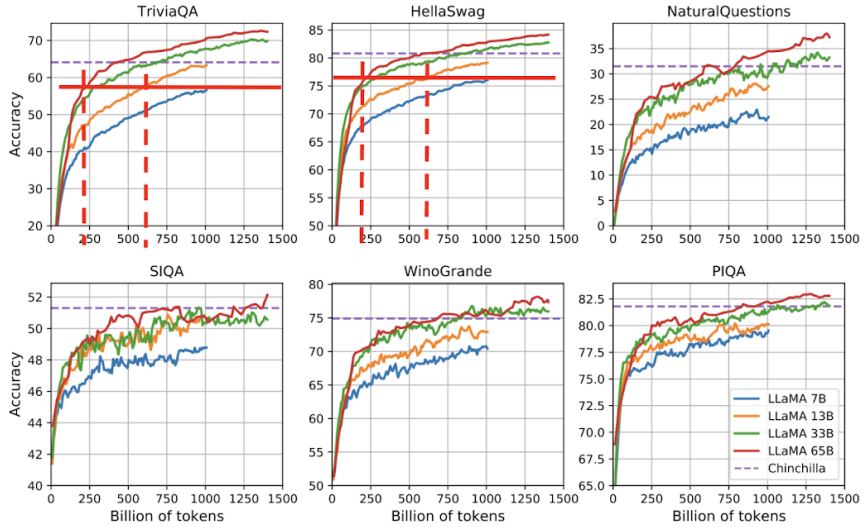


Touvron, Hugo, et al. "Llama: Open and efficient foundation language models." (2023).

Loss predicts performance

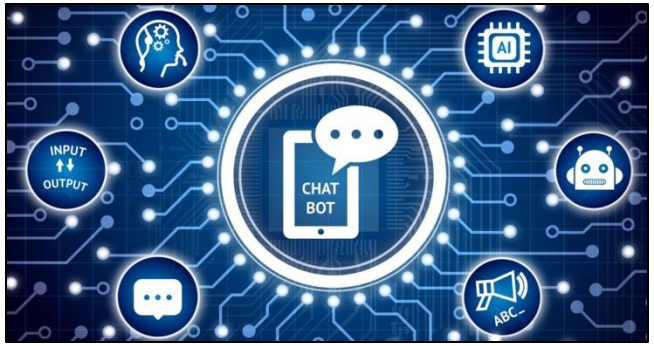
Loss determines downstream performance. Both large and small models have the same performance if they have the same loss.





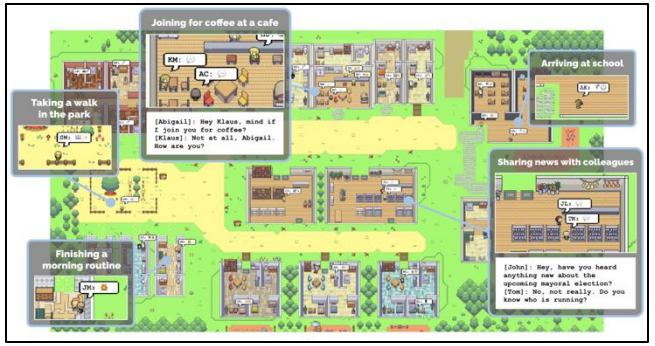
Touvron, Hugo, et al. "Llama: Open and efficient foundation language models." (2023).

LLMs are powerful, but very expensive to deploy



Conversational AI

Content Generation

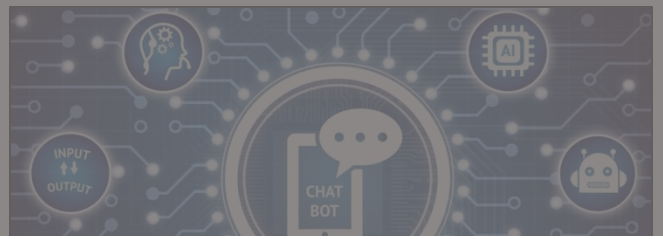


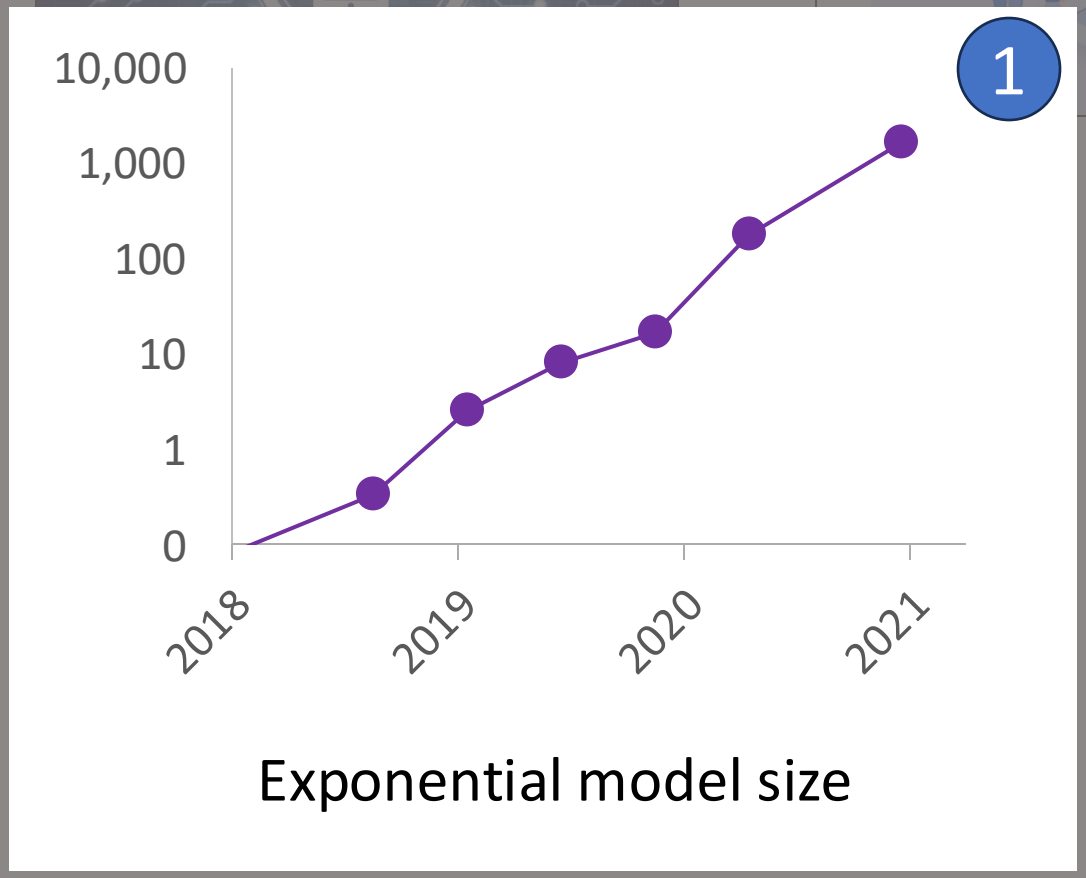
AI Agents

Major Challenges: memory IO (*Pope et al.*) + limited context window

- large mem, e.g. a Llama2-70B model needs
  - 140 GB for parameters,
  - 160 GB for activation (KV cache),
    even with Multi-Group-Attention (8K seqlen + 64 batch size)
- low parallelizability, e.g. generate 100 tokens -> load model, KV cache 100 times
- Perplexity explosion beyond pre-trained windows

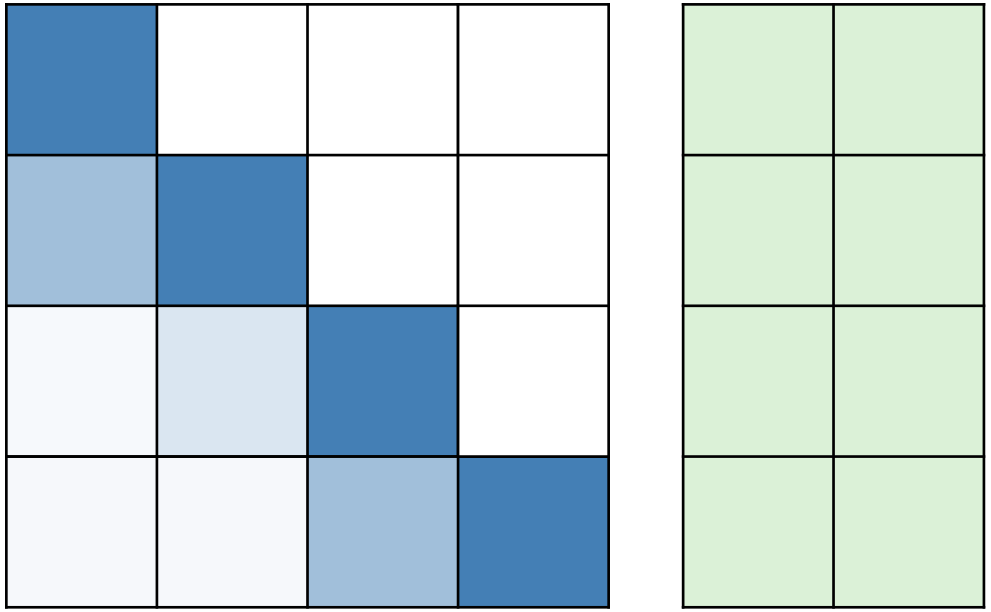
LLMs are Powerful , but Very Expensive to Deploy





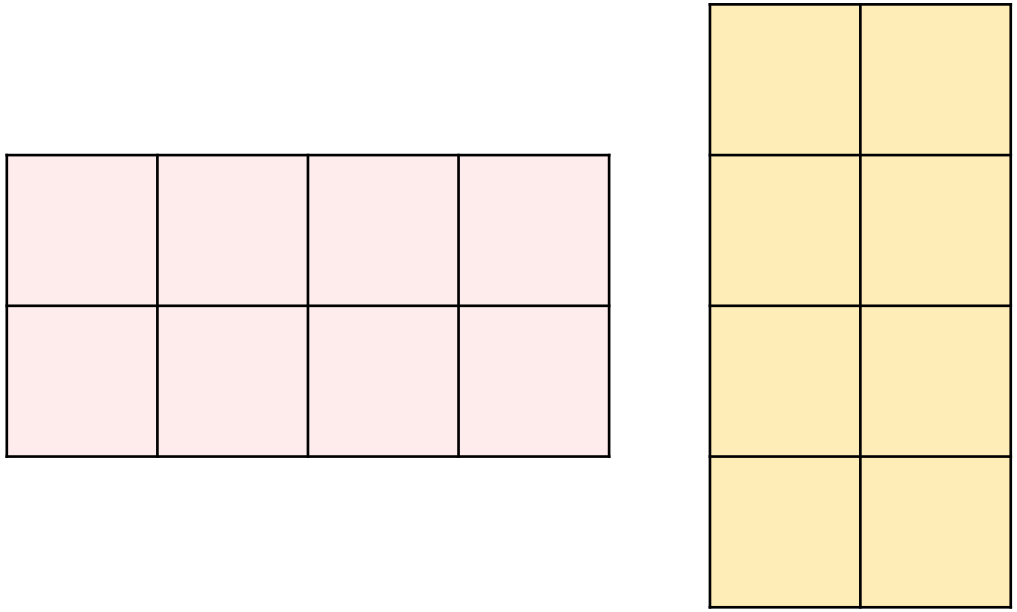
Background: Transformer Architecture

Attention



$\{W_q, W_k, W_v, W_o\ \} \in R^{d\text{x}d}$

MLP



$\{W_1, W_2\} \in R^{d\text{x}4d}$

LLM Static Pruning

SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot

Elias Frantar [1] **Dan Alistarh** [1 2]

Abstract

We show for the first time that large-scale generative pretrained transformer (GPT) family models can be pruned to at least 50% sparsity in *one-shot, without any retraining*, at minimal loss of accuracy. This is achieved via a new pruning method called `SparseGPT`, specifically designed to work efficiently and accurately on massive GPT-family models. We can execute `SparseGPT` on the largest available open-source models, OPT-175B and BLOOM-176B, in under 4.5 hours, and can reach 60% unstructured sparsity with negligible increase in perplexity: remarkably, more than 100 billion weights from these models can be ignored at inference time. `SparseGPT` generalizes to semi-structured (2:4 and 4:8) patterns, and is compatible with weight quantization approaches. The code is available at: https://github.com/IST-DASLab/sparsegpt.

Pruning has a long history (LeCun et al., 1989; Hassibi et al., 1993), and has been applied successfully in the case of vision and smaller-scale language models (Hoefler et al., 2021). Yet, the best-performing pruning methods require *extensive retraining* of the model to recover accuracy. In turn, this is extremely expensive for GPT-scale models. While some accurate *one-shot* pruning methods exist (Hubara et al., 2021a; Frantar et al., 2022b), compressing the model without retraining, unfortunately even they become very expensive when applied to models with billions of parameters. Thus, to date, there is essentially no work on accurate pruning of billion-parameter models.

Overview. In this paper, we propose `SparseGPT`, the first accurate one-shot pruning method which works efficiently at the scale of models with 10-100+ billion parameters. `SparseGPT` works by reducing the pruning problem to a set of extremely large-scale instances of *sparse regression*. It then solves these instances via a new approximate sparse regression solver, which is efficient enough to execute in a few hours on the largest openly-available GPT models (175B parameters), on a single GPU. At the same time, `SparseGPT` is accurate enough to drop negligible accuracy post-pruning,

$$\text{argmin}_{\text{mask } M_\ell, \hat{W}_\ell} \|W_\ell X_\ell - (M_\ell \odot \hat{W}_\ell) X_\ell\|_2^2.$$

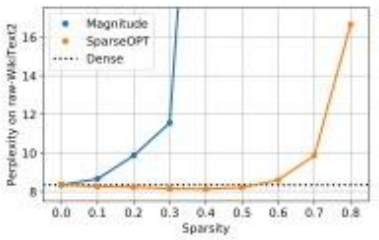


Figure 1. Sparsity-vs-perplexity comparison of `SparseGPT` against magnitude pruning on OPT-175B, when pruning to different *uniform* per-layer sparsities.

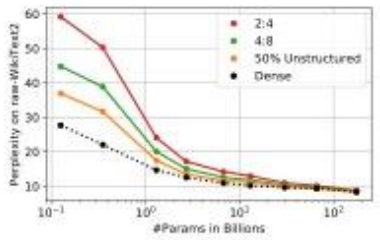


Figure 2. Perplexity vs. model and sparsity type when compressing the entire OPT model family (135M, 350M, ..., 66B, 175B) to different sparsity patterns using `SparseGPT`.

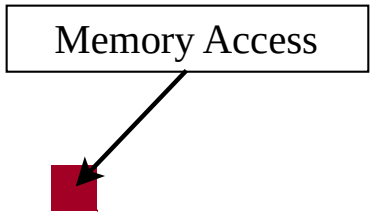
Challenges

The idea sparsity or pruning is not new!

- Long history in ML, statistics, neuroscience, signal processing … *(Lecun et al. 90, Donoho 92, Tibshirani 96, Foldiak et al. 03, Candes et al. 05)*

But hard to speed up sparse LLMs in wall-clock time and maintain quality

- Expensive and infeasible to finetune or retrain
- Difficult to find sparsity that preserves emergent ability of LLMs
- Unstructured sparsity is not hardware-efficient *(Hooker et al. 20)*



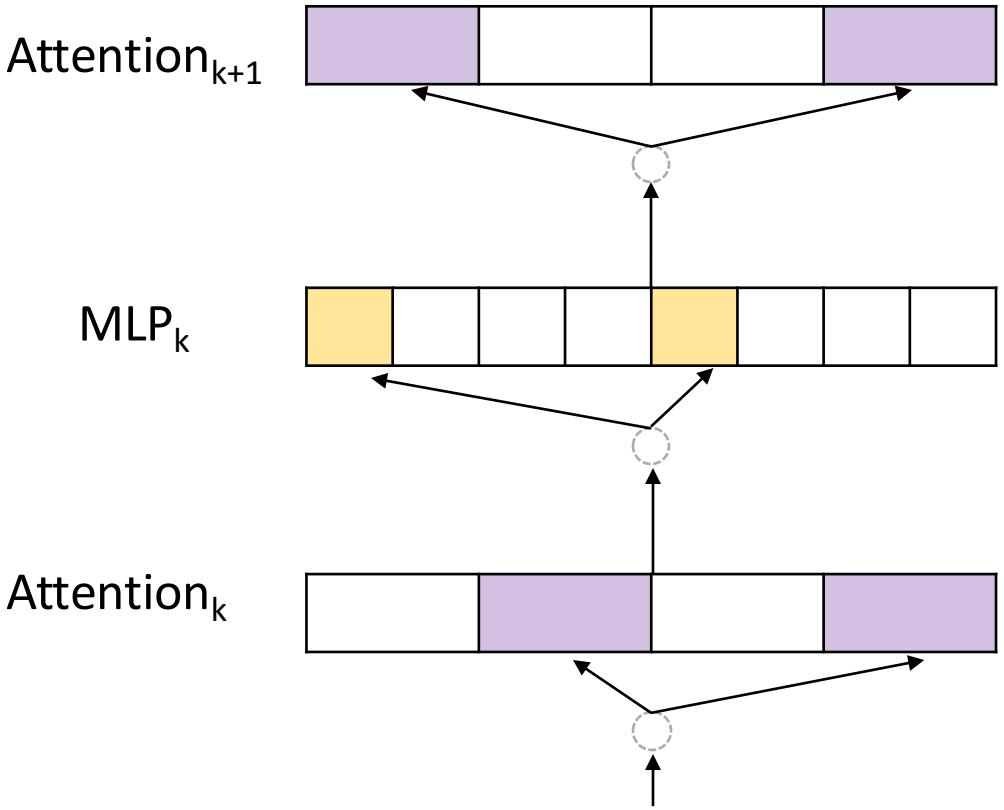
~~Ideal~~ sparsity requires no retraining, maintains quality, and speeds up in wall-clock time.

Contextual

Liu et al. "Deja Vu: Contextual Sparsity for Efficient LLMs at Inference Time." (ICML 2023)

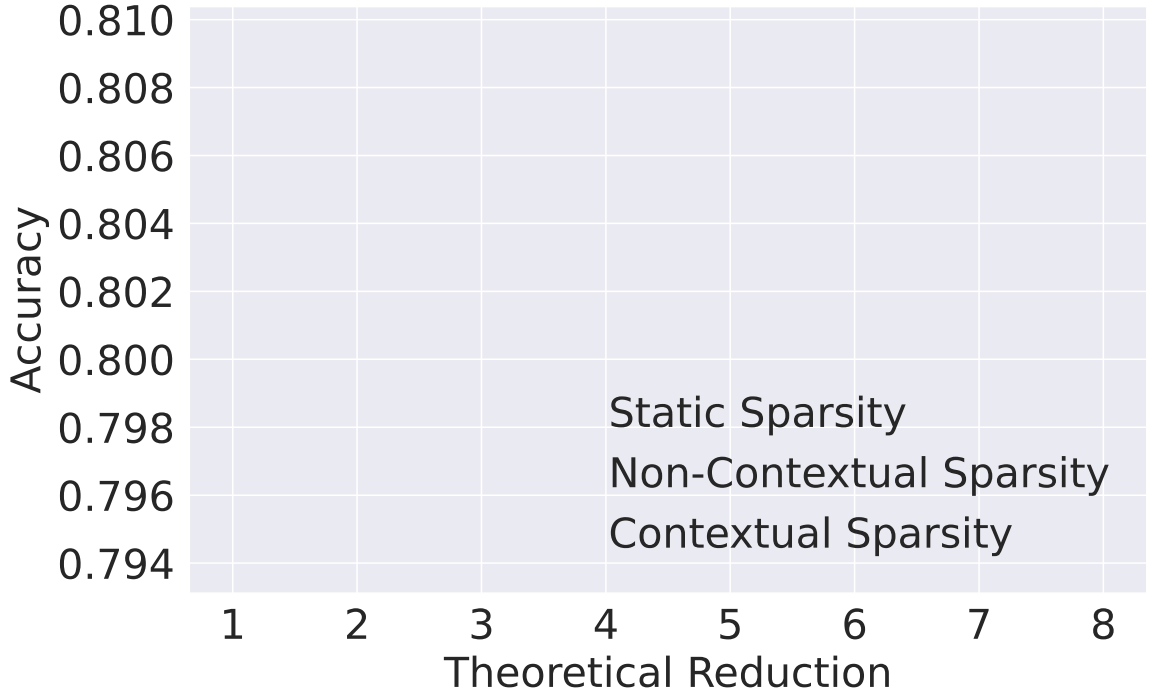
Hypothesis: Contextual Sparsity Exists Given Any Input

Contextual sparsity: small, input-dependent sets of **attention** heads and **MLP** parameters that lead to (approximately) the same output as the full model for an input.



Inspired by:
connections between LLMs, Hidden Markov Models and Viterbi algorithm (Xie et al.)

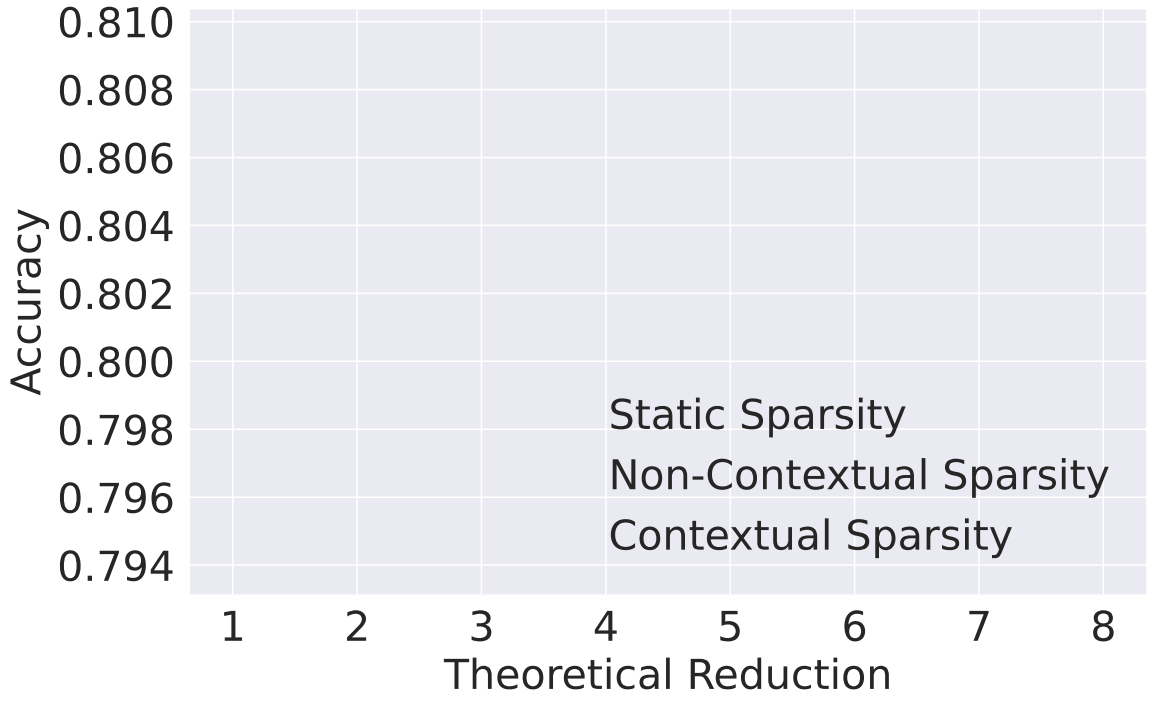
Contextual Sparsity: Existence

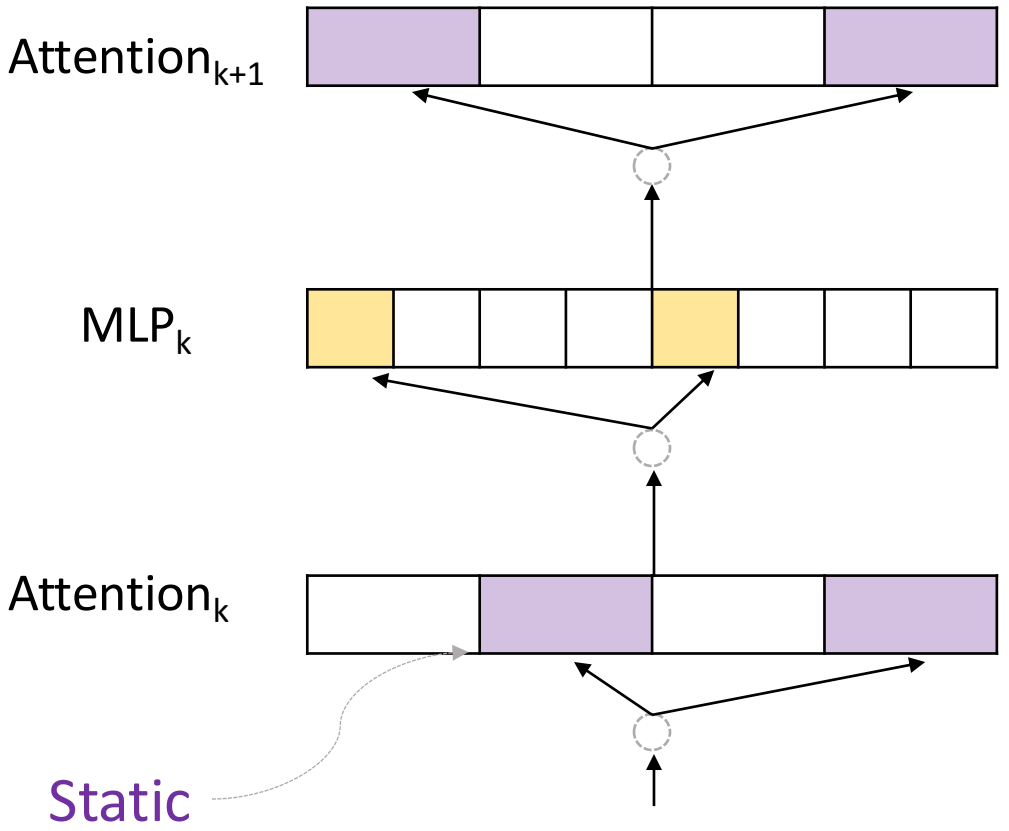


Observation:
keep only high activation in attention/MLP blocks

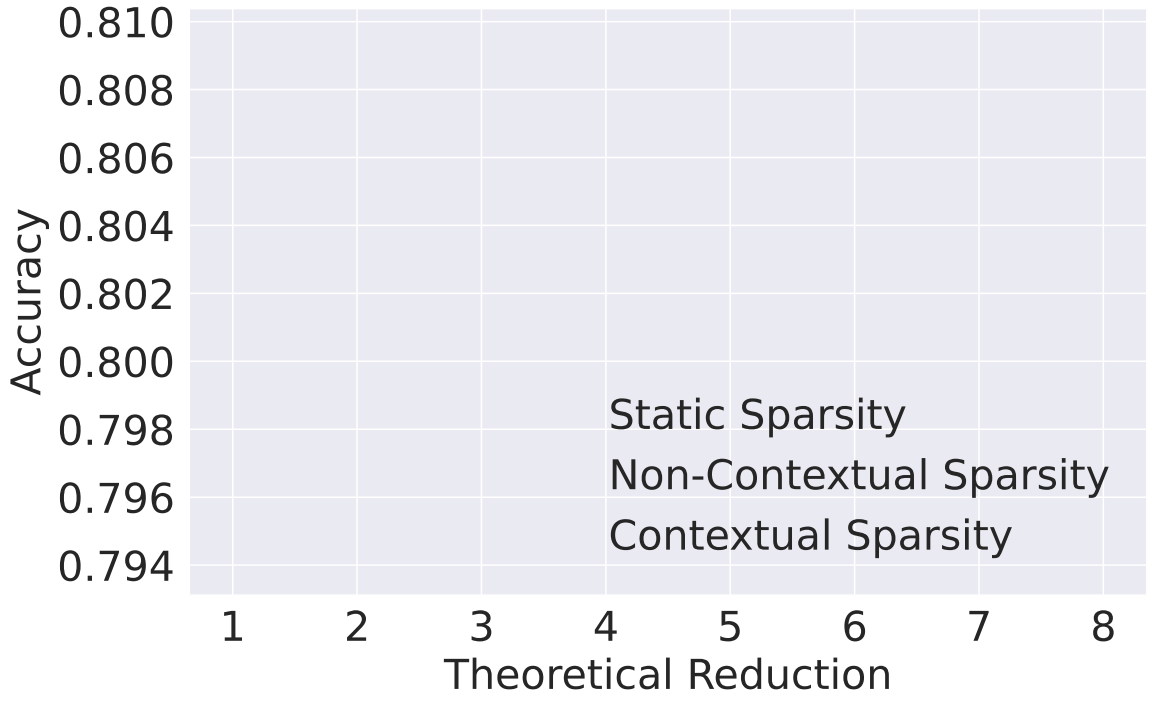
- 85% structured sparse
  - 80% attention, 95% MLP
- lead to 7× potential parameter reduction for each input
- maintain accuracy

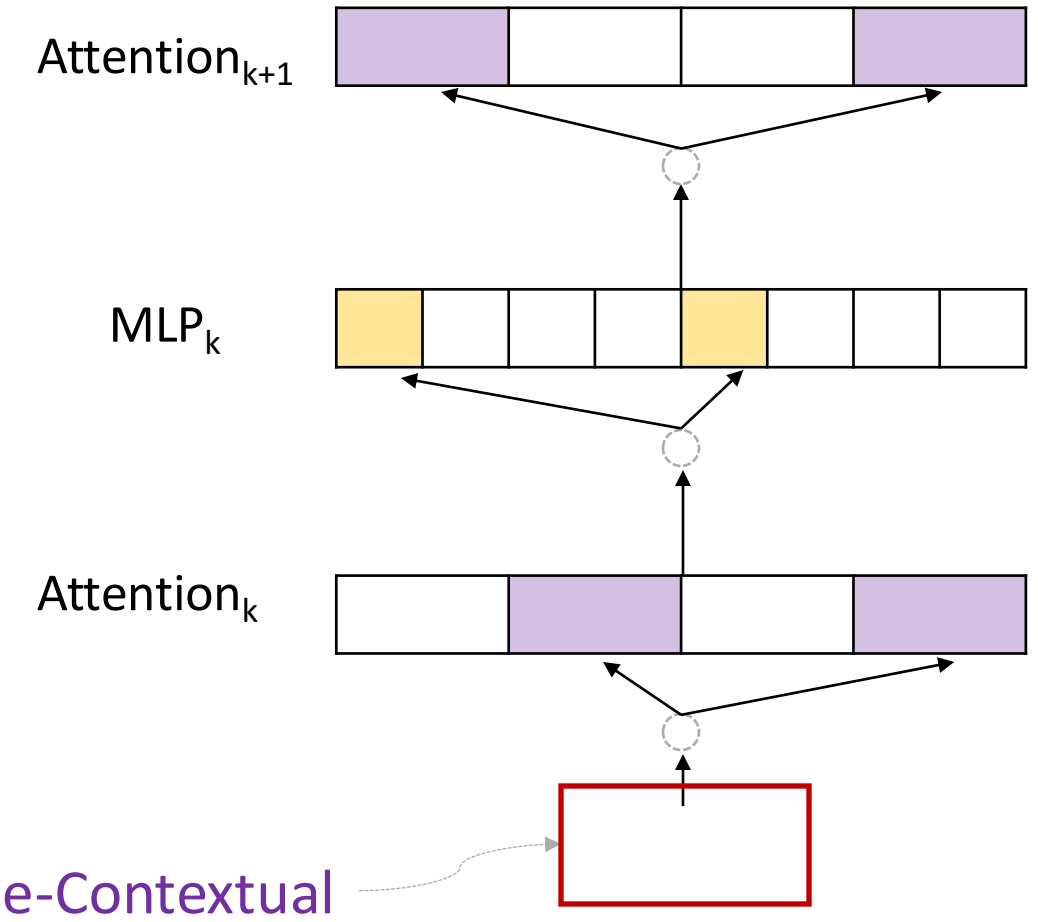
Contextual Sparsity: Existence



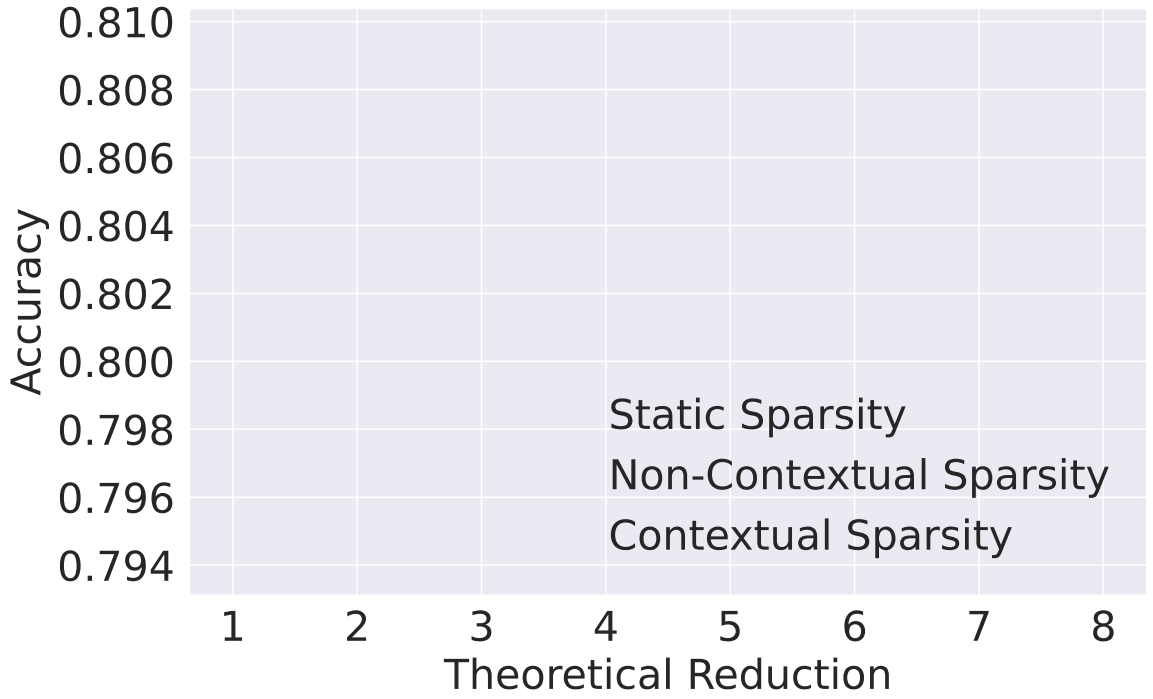


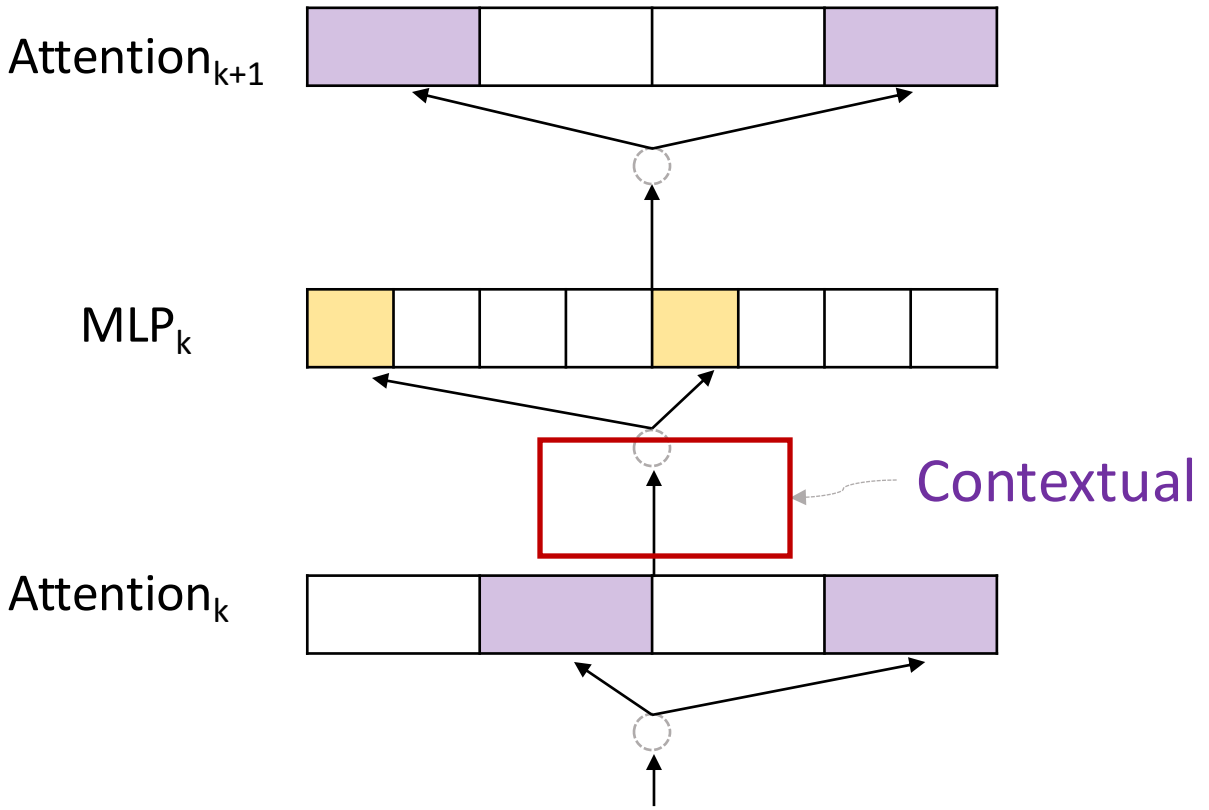
Contextual Sparsity: Existence



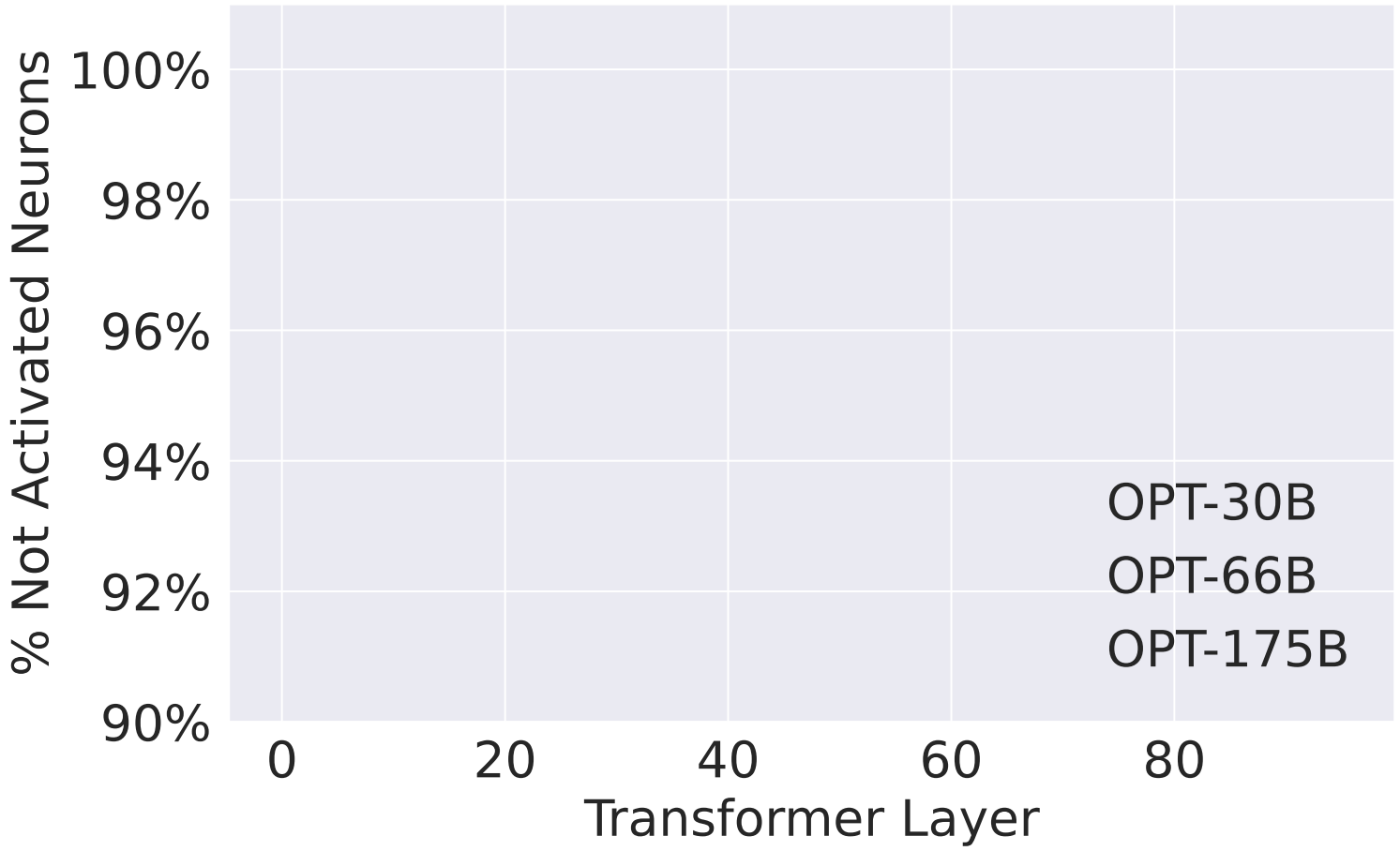


Contextual Sparsity: Existence



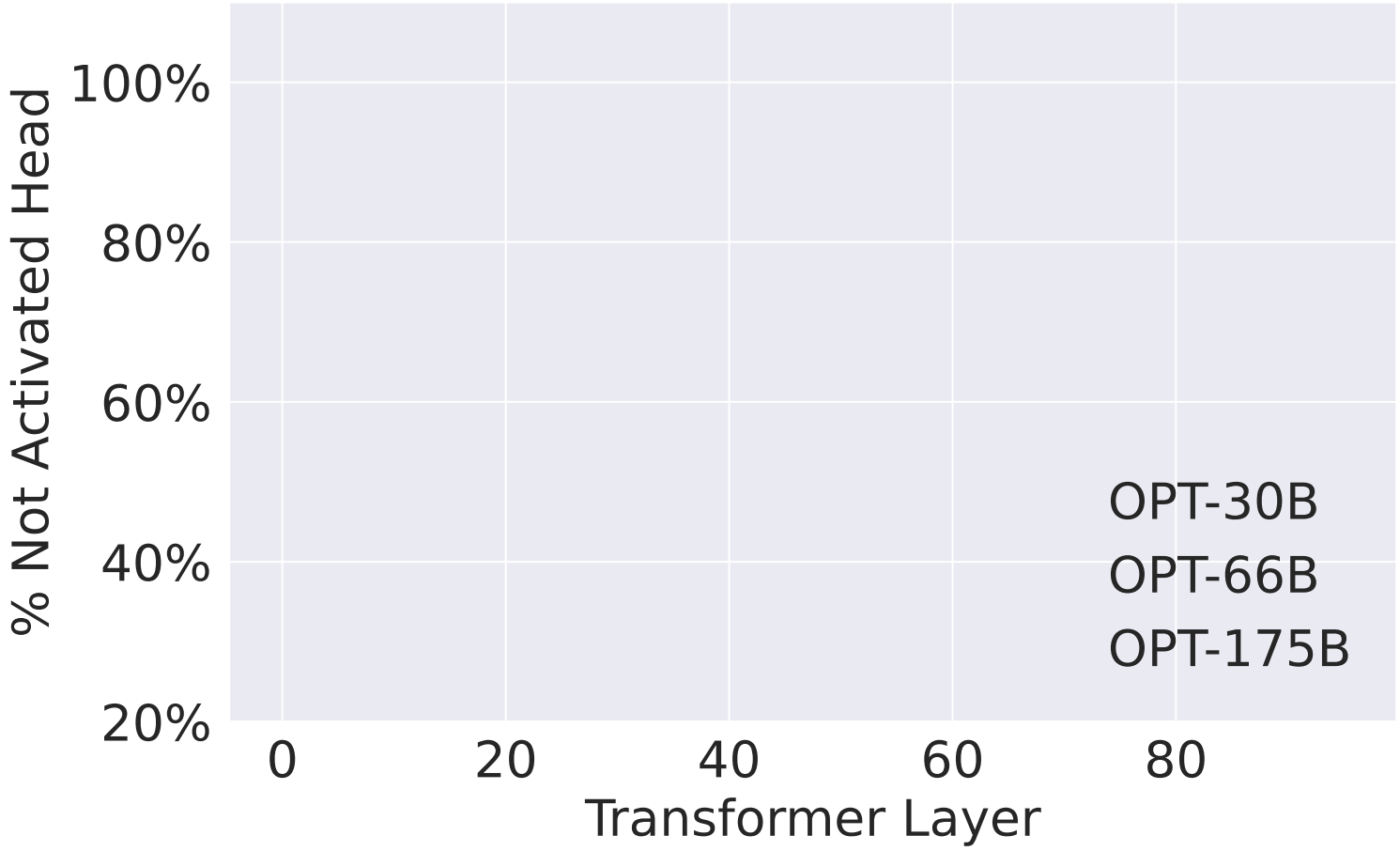


Contextual Sparsity Exists in MLPs

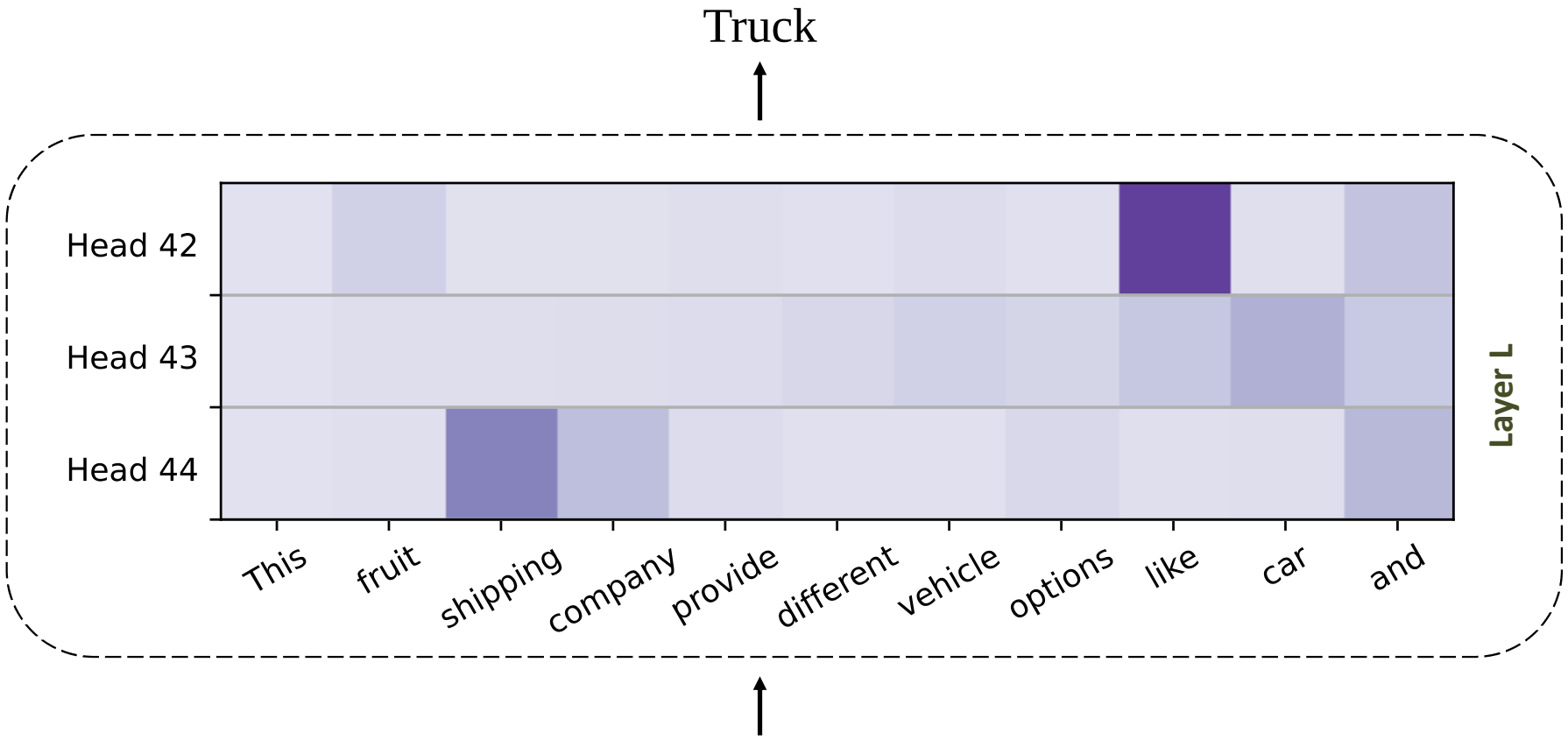


- Due to activation functions, e.g., ReLU, GeLU
- Similar observation in (Li et al.)

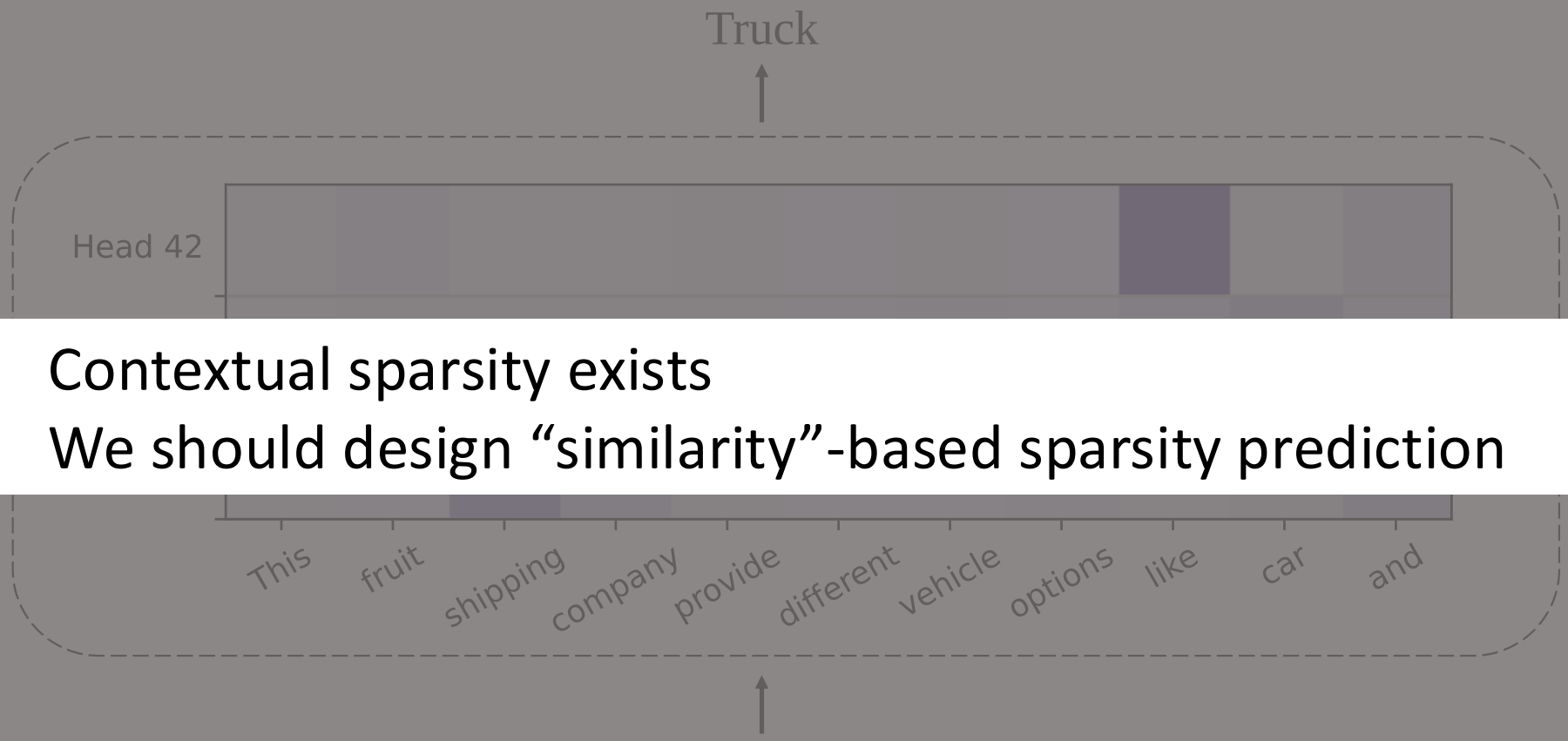
Contextual Sparsity Exists in Attention



Contextual Sparsity Exists in Attention

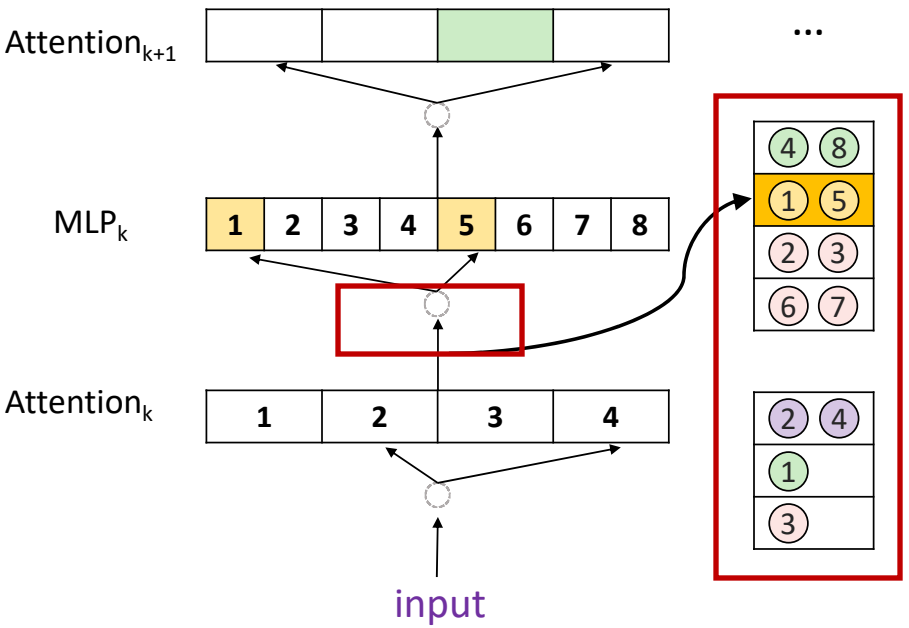


Contextual Sparsity Exists in Attention



- Contextual sparsity exists
- We should design “similarity”-based sparsity prediction

Contextual Sparsity: Prediction



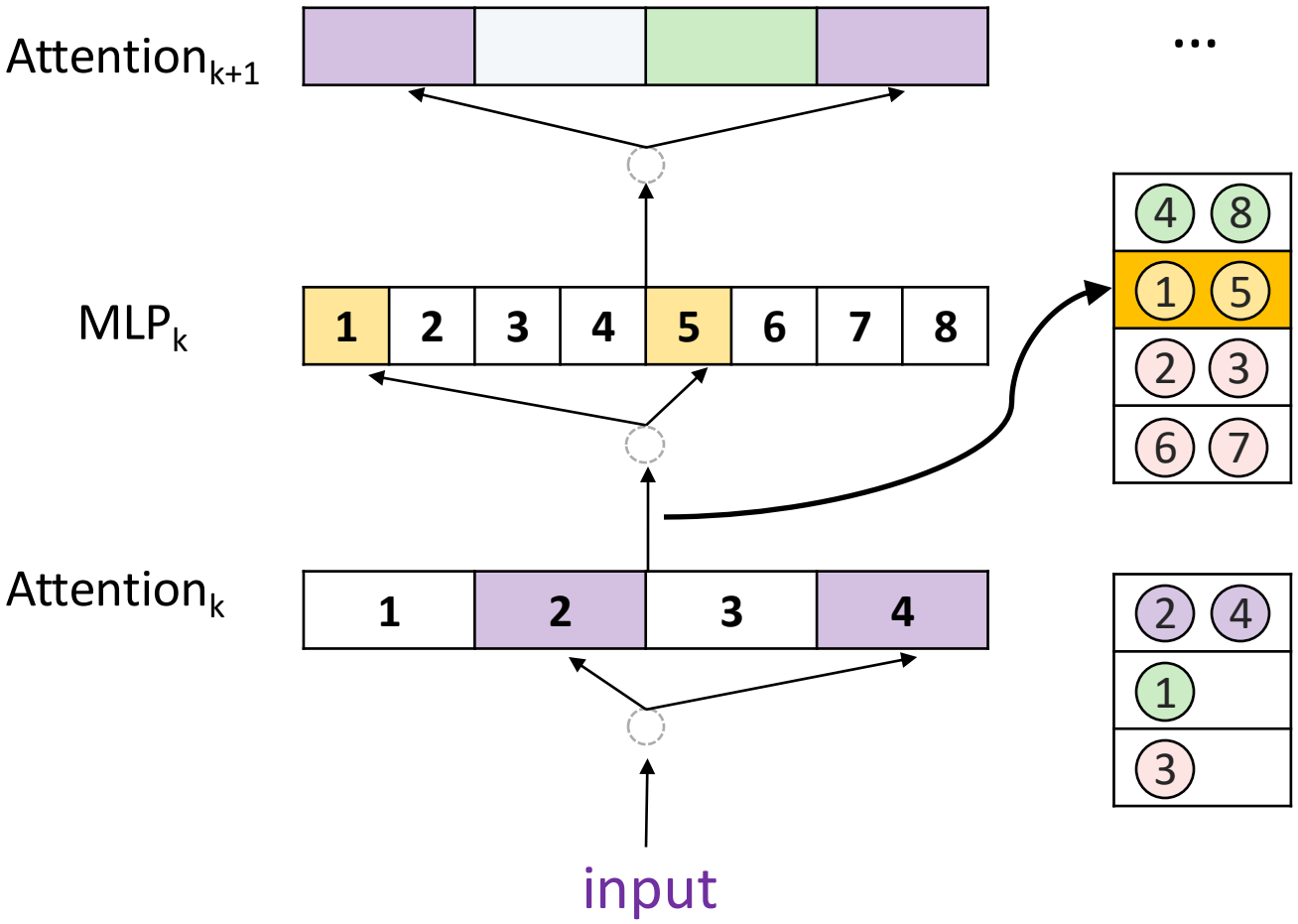
Challenge: how to predict high activation on-the-fly without computing the full attention or MLP?

Key idea: design a "similarity"-based prediction

- formulate the prediction problem as near-neighbor search (NNS).
- Data – neurons or attention heads
- Query – input at each layer

NNS algorithms can make prediction based on the similarity between input & parameters.

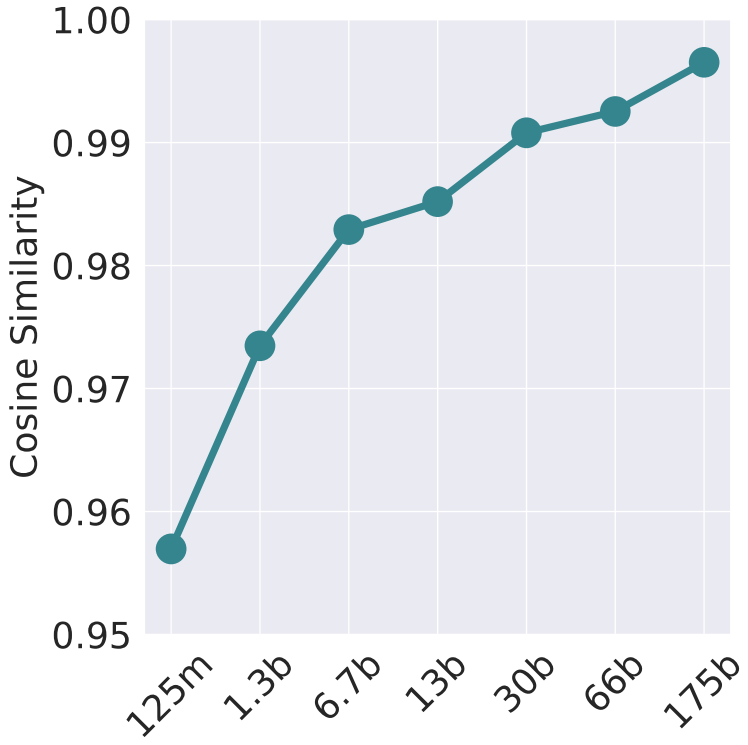
Contextual Sparsity: Efficient

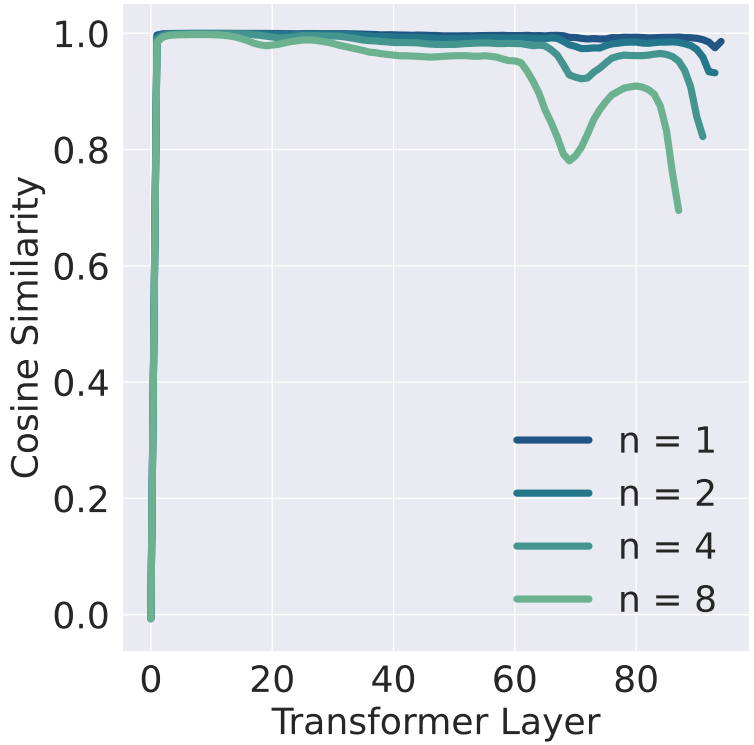


NNS overhead and SpMM

- it performs at each layer
- hash table is not efficient on GPU
- Sparse matmul complicates the implementation

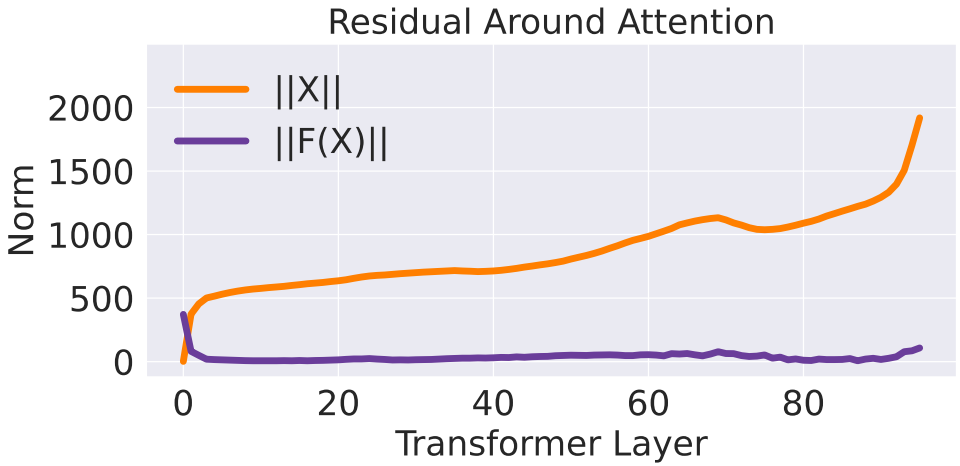
Key Insight: Slowly Changing Embeddings across Layers

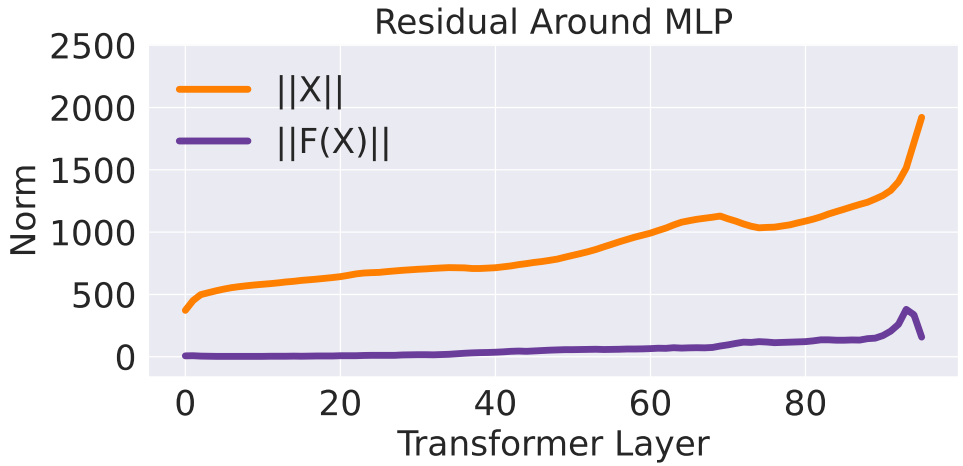




Cosine similarity between representations at consecutive layers is very high.

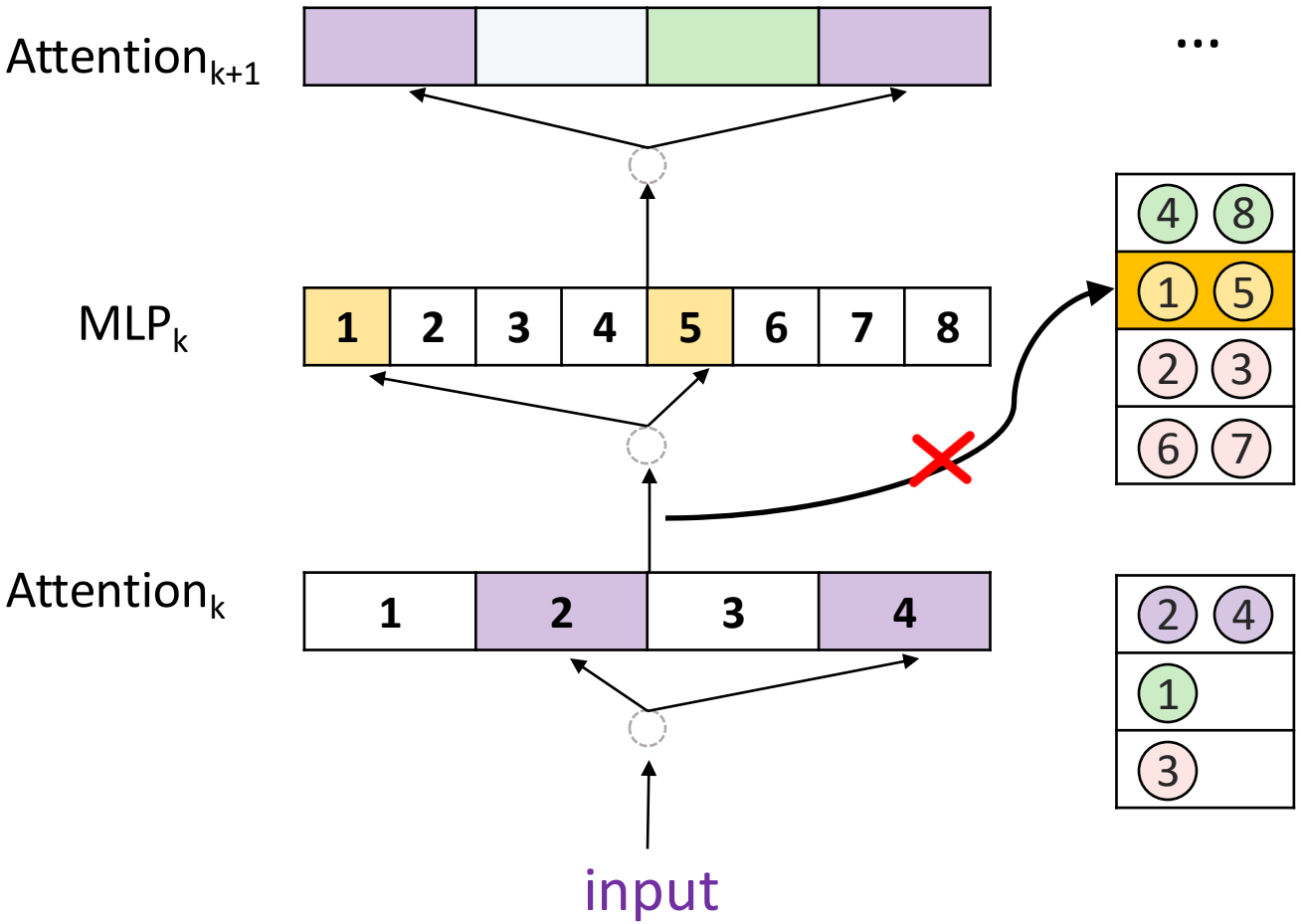
Key Insight: Slowly Changing Embeddings across Layers



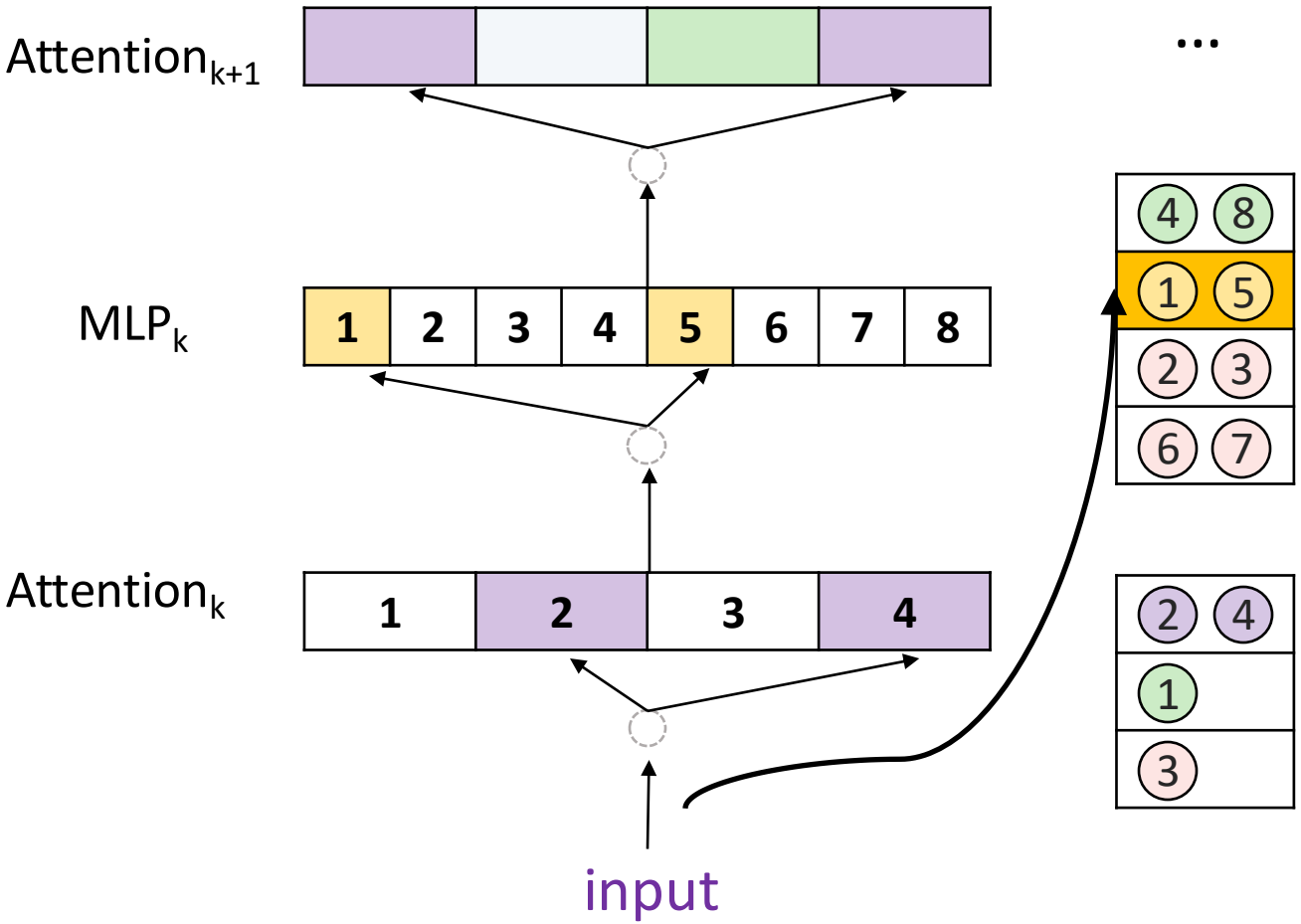


For the residual connection X′ = X +F (X), X's norm dominates.

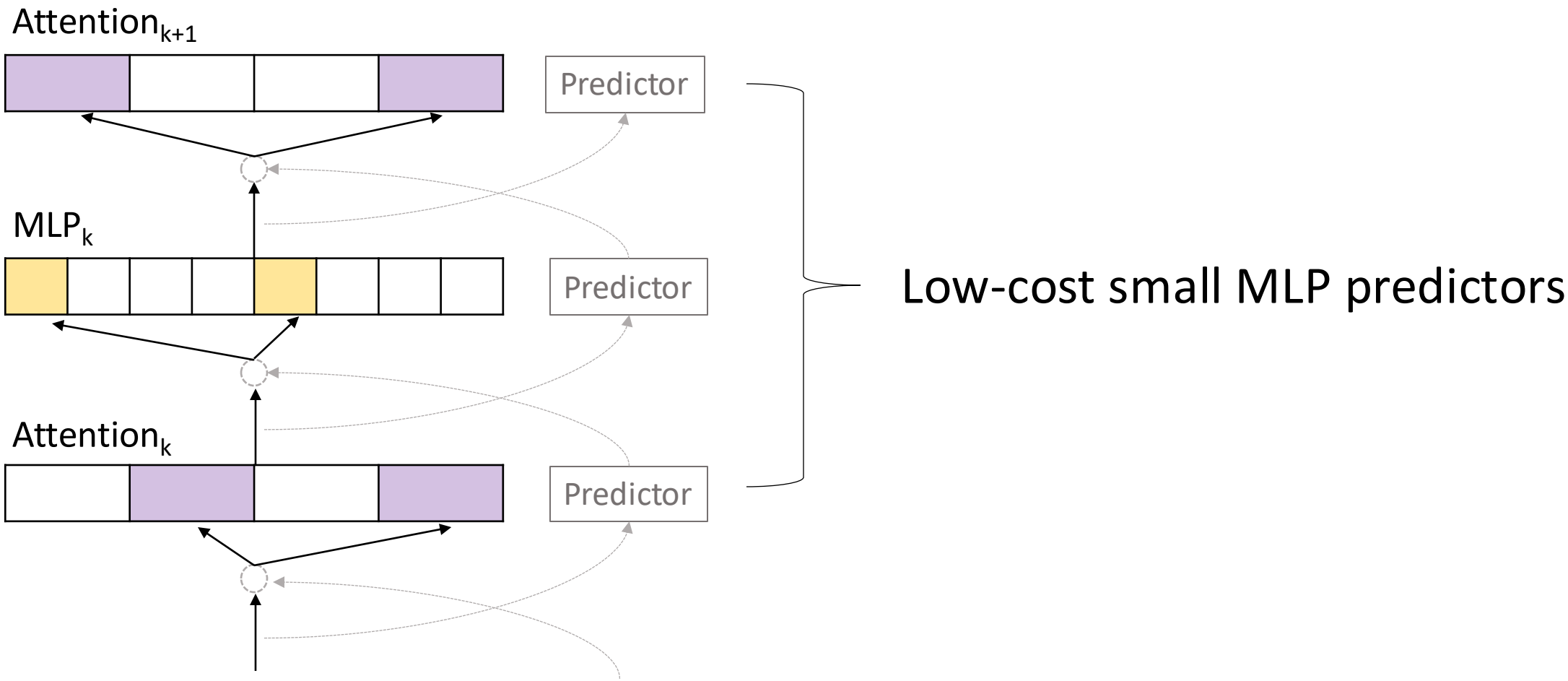
Predict contextual sparsity n-layers ahead



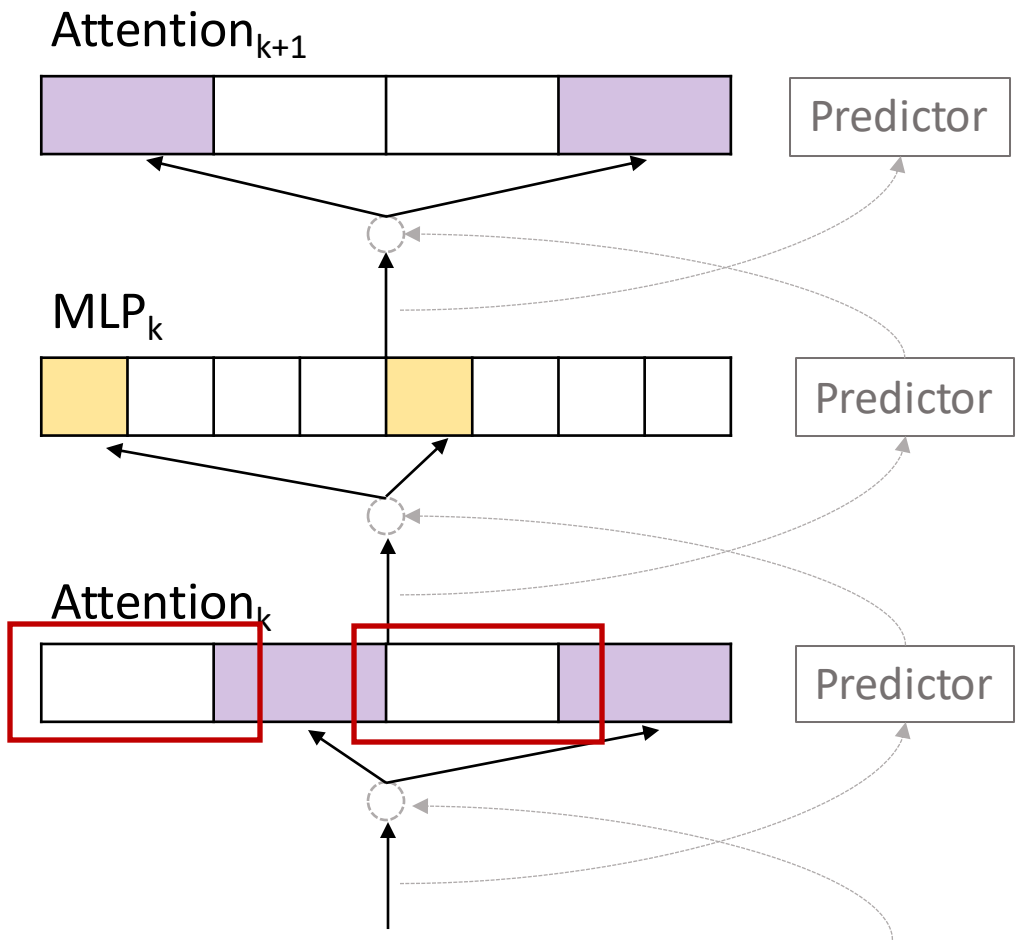
Reduce overhead with Asynchronous Execution



Reduce Overhead with GEMM-based Predictor



Hardware-efficient Implementation



Kernel fusion:

- SpMM, indexing + multiplication
- Triton

Memory coalescing:

- Store Atten out projection and MLP2 in column major

Missing KVCache for a past token:

- latency is bounded by weight loading
- Fill in missing ones when that head is loaded by a future token (compute is free)

Data-Dependent Static Pruning

A Simple and Effective Pruning Approach for Large Language Models

Mingjie Sun[1*] **Zhuang Liu**[2*] **Anna Bair**[1] **J. Zico Kolter**[1,3]
[1]Carnegie Mellon University [2]Meta AI Research [3]Bosch Center for AI

Abstract

As their size increases, Large Languages Models (LLMs) are natural candidates for network pruning methods: approaches that drop a subset of network weights while striving to preserve performance. Existing methods, however, require either retraining, which is rarely affordable for billion-scale LLMs, or solving a weight reconstruction problem reliant on second-order information, which may also be computationally expensive. In this paper, we introduce a novel, straightforward yet effective pruning method, termed Wanda (Pruning by **W**eights **and a**ctivations), designed to induce sparsity in pretrained LLMs. Motivated by the recent observation of emergent large magnitude features in LLMs, our approach prunes weights with the smallest magnitudes multiplied by the corresponding input activations, on a *per-output* basis. Notably, Wanda requires *no* retraining or weight update, and the pruned LLM can be used *as is*. We conduct a thorough evaluation of our method Wanda on LLaMA and LLaMA-2 across various language benchmarks. Wanda significantly outperforms the established baseline of magnitude pruning and performs competitively against recent method involving intensive weight update. Code is available at https://github.com/locuslab/wanda.

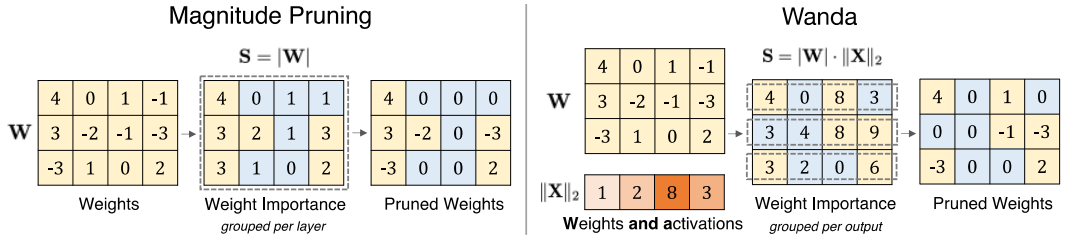


Figure 1: Illustration of our proposed method Wanda (Pruning by **W**eights **and a**ctivations), compared with the magnitude pruning approach. Given a weight matrix $\mathbf{W}$ and input feature activations $\mathbf{X}$, we compute the weight importance as the elementwise product between the weight magnitude and the norm of input activations ($|\mathbf{W}| \cdot \|\mathbf{X}\|_2$). Weight importance scores are compared on a *per-output* basis (within each row in $\mathbf{W}$), rather than globally across the entire matrix.

Method	Weight Update	Calibration Data	Pruning Metric $\mathbf{S}_{ij}$	Complexity
Magnitude	✗	✗	$\lvert\mathbf{W}_{ij}\rvert$	$O(1)$
SparseGPT	✓	✓	$\left[\lvert\mathbf{W}\rvert^2/\mathrm{diag}\left[(\mathbf{X}\mathbf{X}^T+\lambda\mathbf{I})^{-1}\right]\right]_{ij}$	$O(d^3_{\texttt{hidden}})$
Wanda	✗	✓	$\lvert\mathbf{W}_{ij}\rvert \cdot \|\mathbf{X}_j\|_2$	$O(d^2_{\texttt{hidden}})$

Quantization

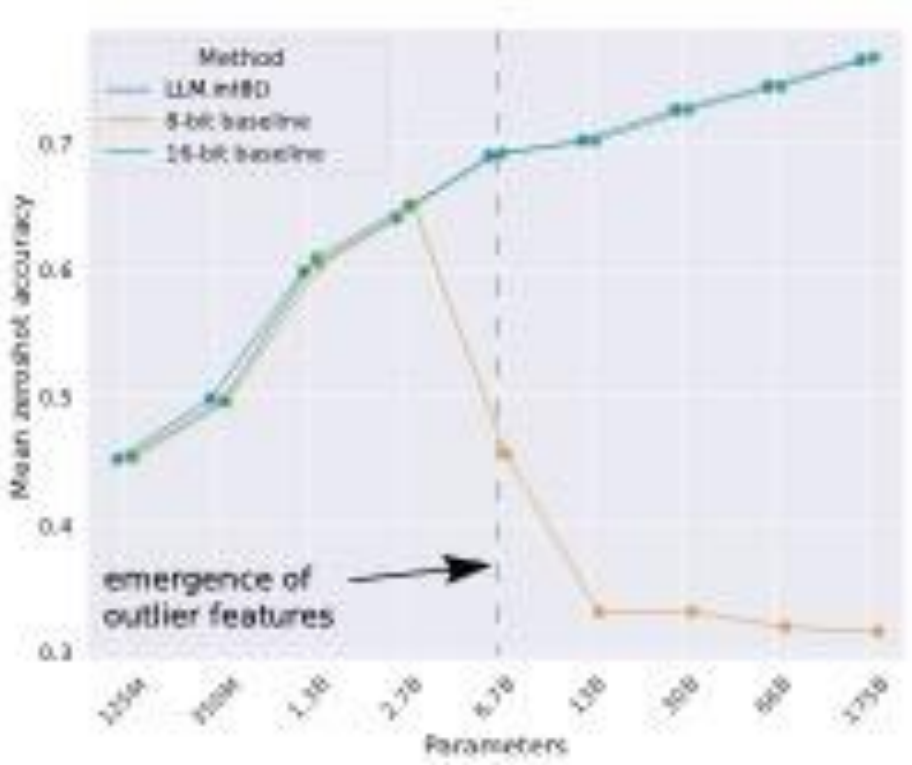
LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale

Tim Dettmers[λ*] **Mike Lewis**[†] **Younes Belkada**[§∓] **Luke Zettlemoyer**[†λ]

University of Washington[λ]
Facebook AI Research[†]
Hugging Face[§]
ENS Paris-Saclay[∓]

Abstract

Large language models have been widely adopted but require significant GPU memory for inference. We develop a procedure for Int8 matrix multiplication for feed-forward and attention projection layers in transformers, which cut the memory needed for inference by half while retaining full precision performance. With our method, a 175B parameter 16/32-bit checkpoint can be loaded, converted to Int8, and used immediately without performance degradation. This is made possible by understanding and working around properties of highly systematic emergent features in transformer language models that dominate attention and transformer predictive performance. To cope with these features, we develop a two-part quantization procedure, **LLM.int8()**. We first use vector-wise quantization with separate normalization constants for each inner product in the matrix multiplication, to quantize most of the features. However, for the emergent outliers, we also include a new mixed-precision decomposition scheme, which isolates the outlier feature dimensions into a 16-bit matrix multiplication while still more than 99.9% of values are multiplied in 8-bit. Using LLM.int8(), we show empirically it is possible to perform inference in LLMs with up to 175B parameters without any performance degradation. This result makes such models much more accessible, for example making it possible to use OPT-175B/BLOOM on a single server with consumer GPUs. We open source our software.



Quantization

SmoothQuant: Accurate and Efficient
Post-Training Quantization for Large Language Models

Guangxuan Xiao[*1] Ji Lin[*1] Mickael Seznec[2] Hao Wu[2] Julien Demouth[2] Song Han[1]
https://github.com/mit-han-lab/smoothquant

Abstract

Large language models (LLMs) show excellent performance but are compute- and memory-intensive. Quantization can reduce memory and accelerate inference. However, existing methods cannot maintain accuracy and hardware efficiency at the same time. We propose SmoothQuant, a training-free, accuracy-preserving, and general-purpose post-training quantization (PTQ) solution to enable 8-bit weight, 8-bit activation (W8A8) quantization for LLMs. Based on the fact that weights are easy to quantize while activations are not, SmoothQuant smooths the activation outliers by offline *migrating* the quantization difficulty from activations to weights with a mathematically equivalent transformation. SmoothQuant enables an INT8 quantization of *both* weights and activations for all the matrix multiplications in LLMs, including OPT, BLOOM, GLM, MT-NLG, Llama-1/2, Falcon, Mistral, and Mixtral models. We demonstrate up to 1.56× speedup and 2× memory reduction for LLMs with negligible loss in accuracy. SmoothQuant enables serving 530B LLM within a single node. Our work offers a turn-key solution that reduces hardware costs and democratizes LLMs.

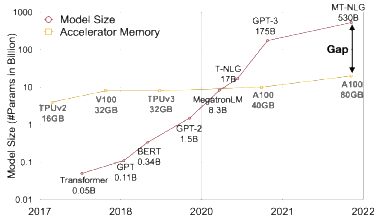
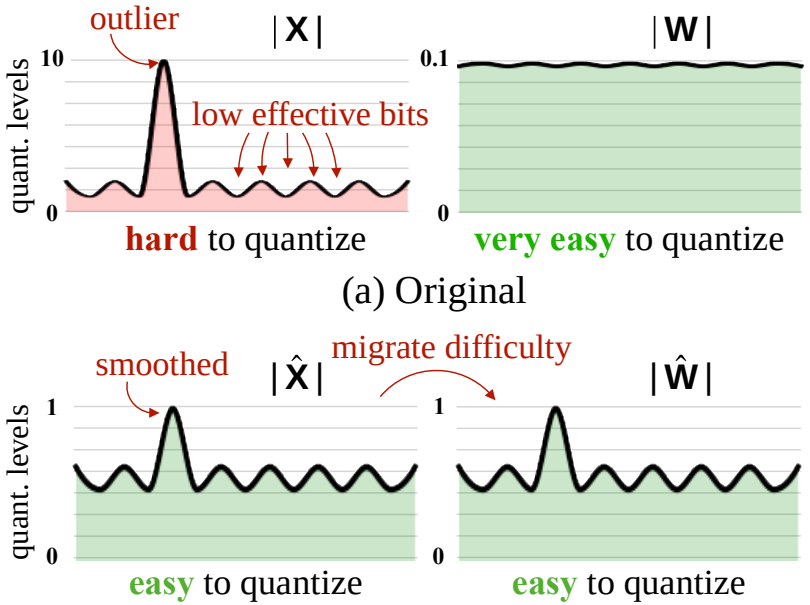


Figure 1: The model size of large language models is developing at a faster pace than the GPU memory in recent years, leading to a big gap between the supply and demand for memory. Quantization and model compression techniques can help bridge the gap.

store and run in FP16, requiring 8→48GB A6000 GPUs or 5→80GB A100 GPUs just for inference. Due to the huge computation and communication overhead, the inference latency may also be unacceptable to real-world applications. *Quantization* is a promising way to reduce the cost of LLMs (Dettmers et al., 2022; Yao et al., 2022). By quantizing the *weights and activations* with low-bit integers, we can reduce GPU memory requirements, in size and bandwidth, and accelerate compute-intensive operations (i.e., GEMM in linear layers, BMM in attention). For instance, INT8



Quantization

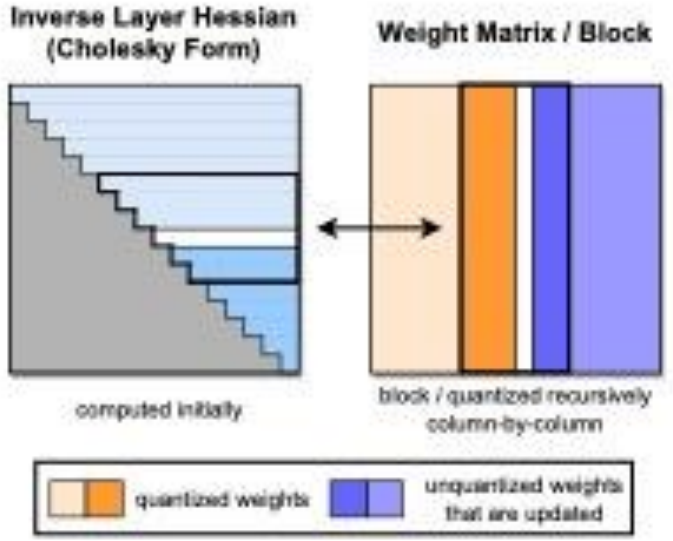
Published as a conference paper at ICLR 2023

GPTQ: ACCURATE POST-TRAINING QUANTIZATION FOR GENERATIVE PRE-TRAINED TRANSFORMERS

Elias Frantar* IST Austria
Saleh Ashkboos ETH Zurich
Torsten Hoefler ETH Zurich
Dan Alistarh IST Austria & NeuralMagic

ABSTRACT

Generative Pre-trained Transformer models, known as GPT or OPT, set themselves apart through breakthrough performance across complex language modelling tasks, but also by their extremely high computational and storage costs. Specifically, due to their massive size, even inference for large, highly-accurate GPT models may require multiple performant GPUs, which limits the usability of such models. While there is emerging work on relieving this pressure via model compression, the applicability and performance of existing compression techniques is limited by the scale and complexity of GPT models. In this paper, we address this challenge, and propose GPTQ, a new one-shot weight quantization method based on approximate second-order information, that is both highly-accurate and highly-efficient. Specifically, GPTQ can quantize GPT models with 175 billion parameters in approximately four GPU hours, reducing the bitwidth down to 3 or 4 bits per weight, with negligible accuracy degradation relative to the uncompressed baseline. Our method more than doubles the compression gains relative to previously-proposed one-shot quantization methods, preserving accuracy, allowing us for the first time to execute an 175 billion-parameter model inside a single GPU for generative inference. Moreover, we also show that our method can still provide reasonable accuracy in the *extreme quantization* regime, in which weights are quantized to 2-bit or even *ternary* quantization levels. We show experimentally that these improvements can be leveraged for end-to-end inference speedups over FP16, of around 3.25x when using high-end GPUs (NVIDIA A100) and 4.5x when using more cost-effective ones (NVIDIA A6000). The implementation is available at `https://github.com/IST-DASLab/gptq`.



$$\text{argmin}_{\widehat{\mathbf{W}}} ||\mathbf{WX} - \widehat{\mathbf{W}}\mathbf{X}||_2^2.$$

Quiz Passcode: Chameleon